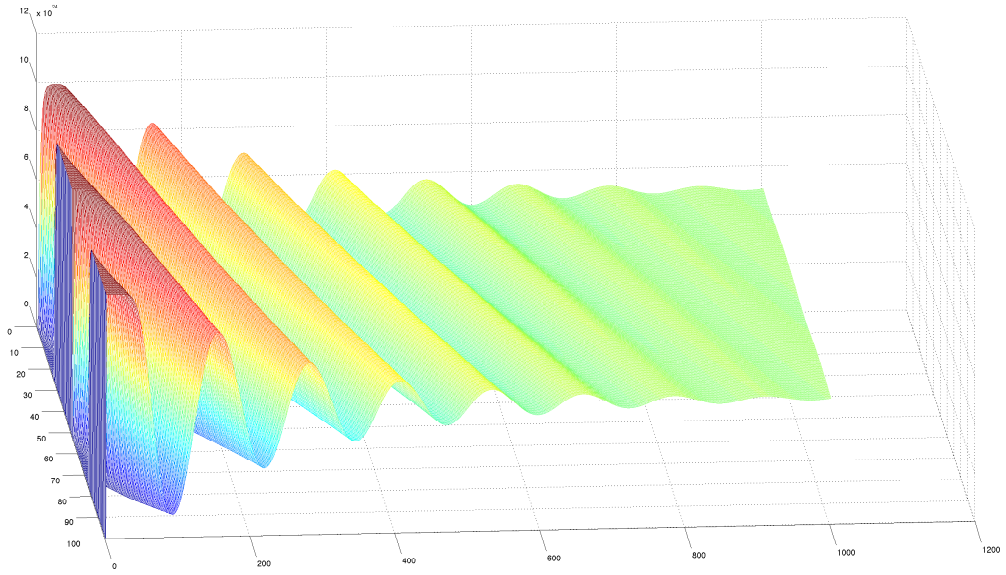
A Study of Possible Applications for Jacobian-Free Newton Krylov Methods in Nuclear Reactor Physics



Dion Koeze
Supervisor: Danny Lathouwers
TU Delft
September 13, 2009

Abstract

A Survey of a Jacobian-free Newton Krylov method using GMRES is presented in this report. The goal of the project is to see whether JFNK has applications in nuclear reactor physics. JFNK methods are methods that solve sets of non-linear equations. The sets of non-linear equations arise from coupled physics problems, as found in nuclear reactor physics. JFNK is especially good at solving coupled sets of equations, since it does not need the Jacobian matrix to solve the problem. It uses an approximation to the Jacobian matrix, this is possible because the Jacobian matrix is only needed as a matrix vector product.

In this report two test problems are used to test the JFNK method. First the heating of a one dimensional rod, while the rod cools by emiting radiation, modeled as a black body. The second test problem is a model of a molten salt reactor.

Since there are many coupled problems in the area of nuclear reactor physics it can be used in many problems. All problems that were encountered in this project have been solved. The harder problems where JFNK could break down were, however, not encountered. Preconditioning is not considered in this project.

Contents

1 Introduction

Since the last century simulations run on computers are of increasing importance in science and engineering. In almost all areas simulations are used as a way of understanding physical problems. There is of course not one algorithm to simulate problems, there is also not one best algorithm. Each new physical model needs a new examination of what algorithm to use. New algorithms are being developed all the time, which results in more and more physical models that can be simulated and with greater resolution.

The calculations in a simulation are usually to solve an equation of motion. For example the Navier-Stokes equation in simulation of flowing fluids or gases, or the heat equation for diffusion problems. The different forms of these equations demand for different algorithms. Equations of the same form can in general be solved by using the same algorithm. Some examples of problems where one equation of motion describes the whole problem are the flow of air around a building in the wind. Or the heat in a computer chip, which is being cooled by an airflow of the fan in the computer. There are also problems where not one but two or more equations of motion are needed to descirbe the problem. This results in more difficult calculations.

These problems are called “coupled” problems, because two different areas of physics and different equations of motion are “coupled” to each other in the problem. Examples of these kind of problems are found in nuclear reactors and weather systems. In weather systems different equations are used to describe the motion of the air, the formation of clouds, the interaction with the oceans, et cetera. In the physics of nuclear reactors different equations are used to describe the neutronics and thermodynamics in a reactor core. Not just with a different physical meaning, but also a different mathematical form of the equation. This makes the problems hard to solve in a reliable and fast manner.

The Jacobian-Free Newton Krylov methods have been around for a couple of decades, but only since the year 2000 it is more used as a solver for physics problems. Before that it was mostly in the domain of the mathematicians, not yet used as an algorithm. Most of the papers about applications of JFNK are from after 2000. There are some books and papers that describe JFNK from before 2000, several different names are in use for the JFNK method.

Two test problems were used in this research. Both are written in Matlab code. The first was the simulation of the heating of a one dimensional rod with Dirichlet boundary conditions. The rod loses energy due to radiation modeled as a black body. The second is a one dimensional simulation of a molten salt reactor.

The first test problem was at first used to find out how to write a code that solves an equation of motion with the JFNK method. The speed of calculation of the JFNK method is compared to other existing solvers. Once the model of the rod worked with the JFNK method, it was used to test how the JFNK method handles coupling problems. This was done by splitting the rod in half and computing the solutions of each half seperately. This model is very easy for the JFNK method, with easy meaning it does not need many Newton and GMRES iterations to solve the problem to the desired tolerance. This gave rise to the second test problem, which is harder to solve.

The second test problem, the molten salt reactor, is a more difficult problem to solve, because the coupling is between different mathematical formulas. As opposed to the coupling in test problem 1, where the coupling was between the same mathematical formulas. A molten salt reactor is a reactor where the fuel is not stationary, like in other reactors. The fuel (often uranium) is dissolved in a salt, which is melted in order to let the fuel move. The fuel-salt mixture is being pumped around a circuit, which consists of a reactor core, a heat exchanger, a pump and the tubing to transport the mixture from and to all the components. The goal of using a molten salt reactor as test problem is not to examine the workings of a molten salt reactor, but to see how the JFNK method behaves when faced with a more difficult problem.

This project was done as a bachelor thesis project for the applied physics bachelor of the University of Technology Delft. The goal of this project was to investigate whether and how the Jacobian-Free Newton Krylov methods can be used in the Physics of Nuclear Reactors group for simulation purposes.

A quick overview of the report is provided here. First the mathematical background of the JFNK method will be explained. In this chapter there are also some “tips and tricks” on how to use JFNK, which were used in the models of the test problems. These tips and tricks are solutions to problems encountered in the test problems. The second part is about the first of the two test problems, the one dimensional rod. The physical and mathematical model of the heating of a one dimensional rod are explained. Then the outcomes of several tests are presented and discussed, the problems ecountered during the making of the models are also mentioned. The third part is about the second of the two test problems, the molten salt reactor. Here again, the physical and mathematical model of the molten salt reactor are explained first. After that the outcomes of several tests are presented and discussed, as well as the problems that arose in the making of this model. The last chapter discusses the conclusions, whether and how JFNK can be used in simulation of nuclear reactors. It also contains information about further research or testing that can be done on the JFNK method.

2 Mathematics and theory of JFNK

2.1 Newton's method

Newton's method is a method that solves non-linear equations iteratively. First there is a description of Newton's method for an scalar function, second there is Newton's method for vector functions.

2.1.1 Scalar functions

Newton's method approximates roots of functions. Therefore if one wants to solve an equation, the equation will have to be rewritten into a form that solving the equation means solving $f(x) = 0$. The function in the graph of Figure 1 is $f(x) = \frac{x^2}{9} + \frac{2}{9}x$, in red. In blue the tangent at $x = 3$ of that function is plotted. We start with a guess $x^0 = 3$, one can see that tracing the tangent to its intesection with the x-axis leads to a better guess of the nearest zero of $f(x)$ than x^0 was. This point closer to the root will be designated x^1, in this case $x^1 = \frac{9}{8}$. The next iteration, which consists of following the tangent of f at $x = x^1$, will give x^2.

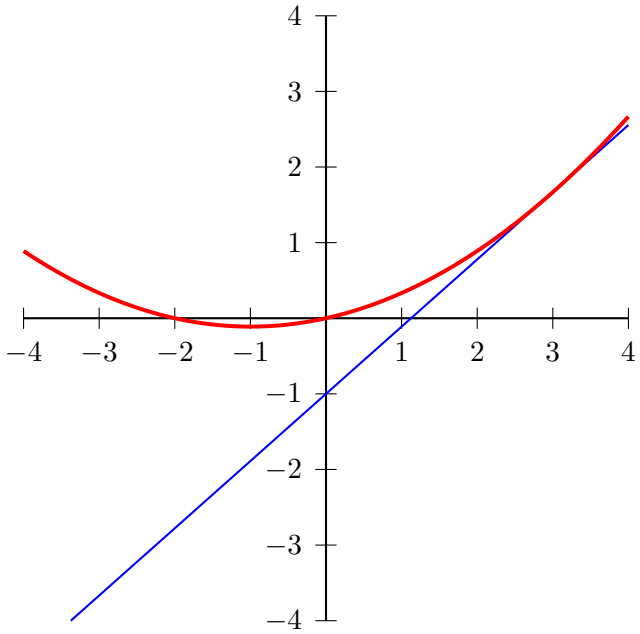


Figure 1: *Illustration of Newton's method. Following back the derivative of the functions leads to a new better guess for x in $f(x) = 0$ ($f(x)$ is in red). Here we start with a guess of $x = 3$, since the tangent (in blue) is at $x = 3$. The better guess here is $\frac{9}{8}$, the intersection of the tangent and the x-axis.*

A general expression for the next guess is

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)} \tag{1}$$

Of course one will in general not acquire the exact solution of the equation $f(x) = 0$, a stop criterion has to be used to determine when the solution is precise enough. Stop criteria will be discussed in the context of Newton's method for vector functions.

The strength of Newton's method can be made visible in a very simple example. Let us determine the square root of 739, which is (exact to the first nine decimal places): 27.184554438. We will have to

use an equation that has as solution the square root of 739, for example $x^2 - 739 = 0$. So we will be finding the root of the function

$$f(x) = x^2 - 739 \tag{2}$$

Below are the solutions of the first couple of iterations with a first guess of $x^0 = 30$, since $\sqrt{900} = 30$, this seems to be a good first guess.

$$\begin{aligned} x^0 &= && = 30 \\ x^1 &= 30 - \tfrac{30^2-739}{2\cdot 739} && = \underline{27}.316666667 \\ x^2 &= 27.31 - \tfrac{27.31^2-739}{2\cdot 27.31} && = \underline{27.184}873907 \\ x^3 &= 27.18 - \tfrac{27.18^2-739}{2\cdot 27.18} && = \underline{27.184554}40 \end{aligned}$$

As one can see the convergence of Newton's method (if the initial guess is close enough to the root is quadratic[5]). This means the number of correct digits roughly doubles every iteration, as is the case in this example: from zero to two to five to nine correct digits.

2.1.2 Vector functions

Now for vector functions, suppose one has a vector function $\mathbf{F}(\mathbf{u})$ and one wants to approximate its roots. This can be achieved by expanding this function by means of a Taylor series, which is

$$\mathbf{F}(\mathbf{u}^{k+1}) = \mathbf{F}(\mathbf{u}^k) + \mathbf{F}'(\mathbf{u}^k)(\mathbf{u}^{k+1} - \mathbf{u}^k) + \mathrm{HOT} \tag{3}$$

Here $\mathbf{u}$ is the state vector of the system and $\mathbf{F}(\mathbf{u})$ is the function obtained by means of the governing equations of the system. The general idea is the same, one follows the tangent to the function at the spot of the guess to the plane perpendicular to the **F**-axis. $\mathbf{u}$ can be compared to x and $\mathbf{F}(\mathbf{u})$ can be compared to $f(x)$ in the scalar version. To obtain a general expression of the next guess there is a short derivation from the Taylor series.

Taking the Taylor series and neglecting the higher order terms (HOT) and setting $\mathbf{F}(\mathbf{u}^{k+1})$ equal to the zero matrix, one obtaines a linear set of equations of the form $\mathbf{Ax} = \mathbf{b}$. From the equation

$$\mathbf{J}(\mathbf{u}^k)\delta\mathbf{u}^k = -\mathbf{F}(\mathbf{u}^k) \quad ; \delta\mathbf{u}^k = \mathbf{u}^{k+1} - \mathbf{u}^k \tag{4}$$

one can calculate the state vector $\mathbf{u}^{k+1}$, which is the next guess. In this equation **J** is the Jacobian matrix of funtion **F**, defined as

$$J_{(i,j)}(\mathbf{u}^k) = \frac{\partial F_i(\mathbf{u}^k)}{\partial u_j} \tag{5}$$

By $J_{(i,j)}$ the element on the ith row and the jth column is meant. **J** is the first derivative of a vector function. One now has a linear equation which can be solved in numerous ways.

For each step in time one has to solve this system until the desired precision is achieved. Various ways of determining the desired precision exist. One of the most commonly used stop critiria is a drop in the norm of the nonlinear residual, which is

$$||\mathbf{F}(\mathbf{u}^k)|| < \tau_r||\mathbf{F}(\mathbf{u}^0)|| + \tau_a \tag{6}$$

In this equation τ_r is the relative tolerance and τ_a is the absolute tolerance. The absolute tolerance is used as to satisfy this criterion when $\mathbf{F}(\mathbf{u}^0)$ is very small [5].

Another stop criterion that is widely used is

$$\frac{||\delta \mathbf{u}^k||}{||\mathbf{u}^k||} < \tau \tag{7}$$

This is the criterion used in all simulations presented in this report. This criterion uses just a relative tolerance τ.

In most situations both criteria result in the same solution, however, there are situations in which a stop criterion can "break down", a situation in which an iteration loop will for example never stop. In all simulations in this report such situations do not occur. Other stop criteria exist, but are not included in this report.

The Newton loop can be sped up by using an extrapolation based on the solution of previous time steps. Before one starts the Newton loop at a certain time step $n+1$, a simple linear extrapolation, like

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \cdot \frac{\mathbf{u}^n - \mathbf{u}^{n-1}}{\Delta t} \tag{8}$$

helps the Newton loop converge faster, since the initial guess is (probably) closer to root of the function **F** than without the extrapolation. This extrapolation simplifies to

$$\mathbf{u}^{n+1} = 2\mathbf{u}^n - \mathbf{u}^{n-1} \tag{9}$$

This extrapolation is actually a first guess of the answer at time $n+1$. Since this guess is a linear extrapolation all higher order effects are neglected. After this guess the Newton loop will correct for all higher order effects, thus finding the answer at time $n+1$.

2.2 Generalized Minimal RESidual method (GMRES)

The GMRES method is used to solve the linear equation obtained from the Newton iteration (eq. 4). In solving this equation the method uses the Krylov subspace. The Krylov subspace is defined as .

$$\mathbf{K} = span(\mathbf{r}_0, \mathbf{J}\mathbf{r}_0, \mathbf{J}^2\mathbf{r}_0, \ldots, \mathbf{J}^{j-1}\mathbf{r}_0) \quad ; \mathbf{r}_0 = -\mathbf{F}(\mathbf{u}) - \mathbf{J}\delta\mathbf{u}_0 \tag{10}$$

wherein j is the size of the square matrix **J**. The algorithm minimalizes the residual in the Krylov subspace. GMRES is not examined in this research, it is just used as a linear solver, because GMRES can be used with JFNK. More information on GMRES can be found in [2] and [1]. More on implementation of GMRES can be found in [9].

The method described so far is known as a Newton Krylov method, for it uses a Newton iteration to approximate the non-linear equation and a Krylov subspace solver (GMRES) to solve the linear equation generated by the Newton iteration step.

The stop criterion used in GMRES is based on $\tau > |\mathbf{b} - \mathbf{A}\mathbf{x}|^2$, in the case of a known matrix **A** (this is the standard norm when solving $\mathbf{A}\mathbf{x} = \mathbf{b}$). The term $\mathbf{A}\mathbf{x}$ is changed into an expression that means the same (the left hand side of a linear equation) but makes use of the Jacobian free approximation, instead of a known matrix **A**.

2.3 Jacobian free approximation

To make it a Jacobian-free Newton Krylov method one has to use the approximation presented in this section. Within the GMRES solver the Jacobian matrix is only needed in a matrix vector product. Therefore one can approximate the Jacobian vector product without computing all of the Jacobian matrix. The product of the Jacobain with a vector **v** is

$$\mathbf{Jv} = \begin{bmatrix} v_1 \frac{\partial F_1}{\partial u_1} + v_2 \frac{\partial F_1}{\partial u_2} \\ v_1 \frac{\partial F_2}{\partial u_1} + v_2 \frac{\partial F_2}{\partial u_2} \end{bmatrix} \tag{11}$$

which is an example for a system with vectors of length two. Note that in the GMRES loop the state vector **u** is the same, as there is no Newton step taken.

To explain the approximation used in JFNK a trivial manipulation of the last equation is used:

$$\mathbf{Jv} = \begin{bmatrix} \frac{F_1(u_1,u_2)+\epsilon v_1 \frac{\partial F_1}{\partial u_1}+\epsilon v_2 \frac{\partial F_1}{\partial u_2}-F_1(u_1,u_2)}{\epsilon} \\ \frac{F_2(u_1,u_2)+\epsilon v_1 \frac{\partial F_2}{\partial u_1}+\epsilon v_2 \frac{\partial F_2}{\partial u_2}-F_2(u_1,u_2)}{\epsilon} \end{bmatrix} \tag{12}$$

In this equation ϵ is a small number, here it is - in most cases - chosen to be the square root of $\epsilon_{machine}$. In Matlab the square root of the machine epsilon on the computer the programs were run on is about $1.5 \cdot 10^{-8}$. The machine epsilon is the smallest distinguishable difference between two numbers on the machine (computer) one is using. This choice of ϵ will not always work! A good value for ϵ is a matter for debate in a lot of problems, however the machine epsilon will work in a lot of cases, more on this can be found in [6].

Part of equation 12 can be seen as the Taylor expansion of the function **F** around **u**, when neglecting the higher order terms. In the first element, $F_1(u_1, u_2) + \epsilon v_1 \frac{\partial F_1}{\partial u_1} + \epsilon v_2 \frac{\partial F_1}{\partial u_2}$, can be seen as the Taylor series of, $F_1(u_1 + \epsilon v_1, u_2 + \epsilon v_2)$, likewise for the second element. Equation 12 then becomes:

$$\mathbf{Jv} \approx \begin{bmatrix} \frac{F_1(u_1+\epsilon v_1,u_2+\epsilon v_2)-F_1(u_1,u_2)}{\epsilon} \\ \frac{F_2(u_1+\epsilon v_1,u_2+\epsilon v_2)-F_2(u_1,u_2)}{\epsilon} \end{bmatrix} \tag{13}$$

Which results in a general equations, for any dimension of the vectors involved:

$$\mathbf{Jv} \approx \frac{\mathbf{F}(\mathbf{u} + \epsilon \mathbf{v}) - \mathbf{F}(\mathbf{u})}{\epsilon} \tag{14}$$

By using this approximation a lot of computing time can be saved, for only a function value has to be evaluated, instead of a matrix vector product. Suppose the system is of size N, then the Jacobian matrix would be size $N \times N$, therefore N^2 values would be evaluated in each Newton iteration. With the JFNK approximation N values will have to be evaluated in each GMRES iteration. Only $\mathbf{F}(\mathbf{u} + \epsilon \mathbf{v})$ changes between GMRES iterations, because **v** changes. $\mathbf{F}(\mathbf{u})$ stays the same and has to be evaluated just once, since **u** does not change during GMRES iterations. For larger values of N this will save a lot of computation time. N should roughly be larger than the average amount of GMRES iterations for JFNK to be quicker.

2.3.1 Example of using JFNK

Now it will become clear why JFNK handles coupled problems well. The function **F** is constructed of the governing equations of the problem. For the sake of clarity, let us use a non-physical example. Take

$$\frac{dx}{dt} = x + y \tag{15}$$

$$\frac{dy}{dt} = xy \tag{16}$$

as governing equations of this example. x and y are arbitrary variables and their equations are found in different areas of physics; it is a coupled problem. Most physical problems have derivatives with respect to time in the governing equations. This means the equations should be discretized in the time (and possibly in the location). In our example the discretized equations are

$$\frac{x^{n+1} - x^n}{\Delta t} = x^{n+1} + y^{n+1} \tag{17}$$

$$\frac{y^{n+1} - y^n}{\Delta t} = x^{n+1} y^{n+1}. \tag{18}$$

in which n denotes the nth step in time. The solution will have to be obtained by letting the Newton iteration converge at every step in time. The vector valued function **F** that belongs to this example is

$$\mathbf{F}(\mathbf{u}^{n+1}) = \begin{bmatrix} x^{n+1} + y^{n+1} - \frac{x^{n+1} - x^n}{\Delta t} \\ x^{n+1} y^{n+1} - \frac{y^{n+1} - y^n}{\Delta t} \end{bmatrix} \tag{19}$$

with the state vector of the system defined as

$$\mathbf{u}^{n+1} = \begin{bmatrix} x^{n+1} \\ y^{n+1} \end{bmatrix} \tag{20}$$

This is the set of equations the Newton iteration will derive a linear set of equations from. The linear set of equations will then be solved by GMRES, whereafter the Newton iteration will take one step closer to one of the roots of **F**. GMRES uses just evaluations of the function **F**, in which the two governing equations can be found. The Jacobian matrix of the two governing equations is not needed.

In general, if one has a physical problem which is made up of N coupled problems, the function **F** will be

$$\mathbf{F}(\mathbf{u}^k) = \begin{bmatrix} \mathbf{F}_1(\mathbf{u}^k) \\ \mathbf{F}_2(\mathbf{u}^k) \\ \vdots \\ \mathbf{F}_N(\mathbf{u}^k) \end{bmatrix} \tag{21}$$

with $\mathbf{F}_i$ the functions of one of the coupled problems. The coupling between the problems is not needed, while that would be needed to construct the Jacobian matrix. This is what makes JFNK well suited for coupled problems.

2.3.2 Scaling

Scaling can also be explained with this example. Scaling means using relative variables, as to make sure every element of the function **F** has roughly the same order of magnitude. Scaling can be necessary when variables with large differences in their order of magnitude are involved, since the approximation in equation 14 will not work. In this example the variables x and y will be scaled as

$$x^* = \frac{x}{x_{ref}} \tag{22}$$

$$y^* = \frac{y}{y_{ref}} \tag{23}$$

where x^* and y^* are the scaled variables of order 1 [10]. The discretized equations (eq. 17 and eq. 18) of our example can be written as

$$\frac{x^{*,n+1} - x^{*,n}}{\Delta t} = x^{*,n+1} + y^{*,n+1} \cdot \frac{y_{ref}}{x_{ref}} \tag{24}$$

$$\frac{y^{*,n+1} - y^{*,n}}{\Delta t} = x^{*,n+1} y^{*,n+1} \cdot x_{ref}. \tag{25}$$

Using these equations to make the function **F**, all elements of **F** are of order $\frac{1}{\Delta t}$, since the derivative term is of that order.

Scaling is important because round off errors may occur when it is not used. The round off errors will occur in the difference of the two evaluations of the approximation of the Jacobian matrix vector product (equation 14). When **u** is much larger than ϵ**v**, the sum of the two will be just **u**. This happens for example when (for some element of **u** and **v**) $u = 2.345 \cdot 10^{32}$ and $\epsilon v = 5.678 \cdot 10^{20}$. Because these two numbers are floating point numbers their sum will be $u + \epsilon v = 2.345 \cdot 10^{32} = u$. To avoid this, scaling is used in this report, this makes sure **v** is of the right order of magnitude. Because **v** is calculated using evaluations of **F** and these evaluations are always of order unity.

In this project the reference value of a variable is always chosen to be the initial value of the variable. Therefore V^* with V any variable is always $V^* = 1$ as initial value.

2.4 Overview of the algorithm

In this section an overview of the whole algorithm is provided.

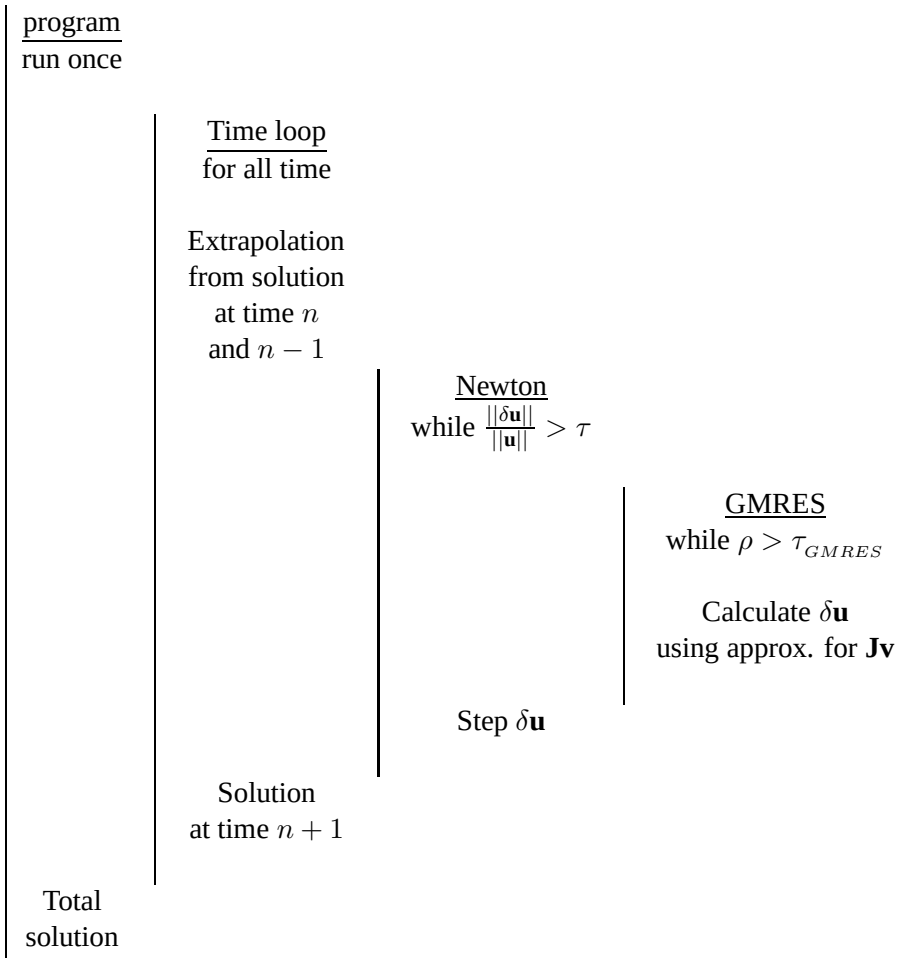


Figure 2: *Overview of the JFNK algorithm used in this report.*

2.5 Advantages and disadvantages of JFNK

In this section a discussion of the advantages and disadvantages of the JFNK method is provided. First the advantages are explained, then the disadvantages.

The JFNK method needs less calculations for large problems, or in other words, problems with many degrees of freedom. It can solve these problems faster than ordinary algorithms. In this project it was about twice as fast as ordinary algorithms, but it is known to be ten or even more times as fast as ordinary algorithms [7].

Not only simple uncoupled problems are solved quicker, it can also solve problems without constructing the Jacobian matrix. This means it can solve coupled problems that cannot be solved by algorithms that need the Jacobian matrix. Many problems in nuclear reactor physics problems and problems with flow of fluids involved can now be solved by JFNK.

A great advantage of JFNK to ordinary methods is the measure for the error of the solution. Ordinary methods involve converging the same part of the problem more than once, after other parts are converged.

The measure for the error is lost in this process and must be reintroduced to have such a measure using ordinary techniques. JFNK always has a measure for the error. This is where the stop criterion of the loop is based upon.

The disadvantages of JFNK will be summarized next. There is really just one disadvantage of JFNK, but this can disrupt the calculations in more than one way. Since only an approximation of the Jacobian matrix is used errors in these calculations can occur. In this project two situations where JFNK breaks down were encountered.

First the approximation can break down when large differences occur in the variables. This problem is described in the mathematical theory section. It can be (partly) solved by using scaled variables. This can still go wrong when variables increase or decrease too much in the solution of the problem.

Even with scaling the approximation can still give wrong answers. This was encountered when a large change occured in one of the input variables. JFNK found a root of the function **F**, but it was not a root that produces a physical realizable solution of the problem. So with a sudden change in input one has to check whether the solution is plausible.

3 Test problem 1: Heating a 1-D rod

3.1 Mathematical and computational model

The first test problem used to explore the JFNK method consists of a one dimensional rod with Dirichlet boundary value conditions. The initial condition is zero Kelvin in the whole rod. The rod loses thermal energy because of radiation, modeled as a black body. Figure 3 gives an overview of the situation. The two temperatures at the ends are given and ϕ_q denotes the flux of radiation energy from the rod. This is chosen as a test problem because the solutions are intuitively clear and the mathematics of the problem is not difficult.

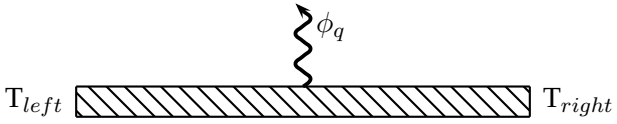


Figure 3: *Schematic overview of the test problem, a one dimensional rod. The rod is heated from the sides, which are Dirichlet boundary conditions. It also loses heat through radiation, modeled as a black body. The initial condition is zero Kelvin in the whole rod.*

The governing equation is the heat equation,

$$\frac{\partial u}{\partial t} = k\frac{\partial^2 u}{\partial x^2} - \sigma u^4 \tag{26}$$

in which u is the temperature, k is the thermal conductivity (in this case $k = 10^2 \frac{\mathrm{W}}{\mathrm{mK}}$) and $\sigma = 5.67 \cdot 10^{-8} \frac{\mathrm{W}}{\mathrm{m^2K^4}}$ is the Stefan-Boltzmann constant. The length of the rod is one meter.

In order to use this equation in iterative methods, it has to be discretized. The discretization in space is done by a finite difference approximation, the discretization in time is done by a backward Euler discretization. The discretized heat equation then becomes

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = k\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2} - \sigma(u_i^{n+1})^4 \tag{27}$$

where $n+1$ is the current moment in time, i is the space discretization index, Δt is the time step and Δx is the spatial step. The rod is divided into $N = \frac{L}{\Delta x}$ segments. This equation has to be solved at every instance in time for every point in the rod.

3.2 Three methods

In order to compare different methods of solving the problem several different approaches were used in the first test problem. Most of the comparison is centered on the difference between the Jacobian vector product approximation and building the Jacobian explicitly. Here the three different versions of the algorithm are presented.

3.2.1 Method I

The first method solves the problem only by means of a Newton iteration. The linear system that needs to be solved in each Newton iteration is solved using a direct solver, which needs the full Jacobian matrix. This program is not so much used to compare other programs to, more so it was used as practice to write the programs for the other two methods.

3.2.2 Method II

The second method uses the Newton method but does not use a direct solver to solve the linear system, instead it uses a GMRES code obtained from C.T. Kelley [5]. In order to use GMRES, the program still needs to calculate the full Jacobian matrix, like in method I.

3.2.3 Method III

The third method is essentially the same as the second program, only in the GMRES code the Jacobian vector product is appromixated by equation 14, this is the JFNK method. This method is of interest in this report.

3.3 Performance of the methods II and III

Method I is left out of this comparison because it is more interesting to see the performance of GMRES with the whole Jacobian versus GMRES with the Jacobian approximation, that is, method II versus method III. Method I uses far more time to compute the answer. For this comparison the boundary conditions

$$T_{left}(n) = T\sin(10\frac{\Delta t}{t}n + 1) + T \tag{28}$$

$$T_{right}(n) = T\sin(5\frac{\Delta t}{t}n + 3) + T \tag{29}$$

are used, where $T = 1000\text{K}$ is some constant temperature. The maximum temperature is $2T = 2000\text{K}$. The initial condition is zero Kelvin throughout the rod. The solution to this problem is found in Figure 4.

Figure 5 shows the computation time of the two methods as a function of the relative tolerance of GMRES and of the Newton iteration. Matlab does not provide a way of counting floating point operations (flops), therefore real computation time was used to show the effectiveness of the methods. On the z axis the relative computation time of the methods is shown, the time of method II is divided by the time of method III. So a value above 1 means method III was quicker, a value below 1 means method II was quicker. In practice tolerances of between 10^{-2} and 10^{-6} are used, in this region method III is clearly quicker than method II. This means the JFNK method is quicker than building the Jacobian matrix.

For comparison of methods II and III the number of Newton iterations and the average number of GMRES iterations per Newton iteration are plotted. The solution is plotted above the iteration count plot to show at what instance in time more iterations are needed to achieve the desired precision. Figure 6(a) shows the iterations of the GMRES method with the full Jacobian (method II), Figure 6(b) shows the iterations of the GMRES method with the Jacobian approximation (method III). Both were made using a relative tolerance in the Newton iteration of 10^{-3} and a tolerance in the GMRES iterations of 10^{-5}.

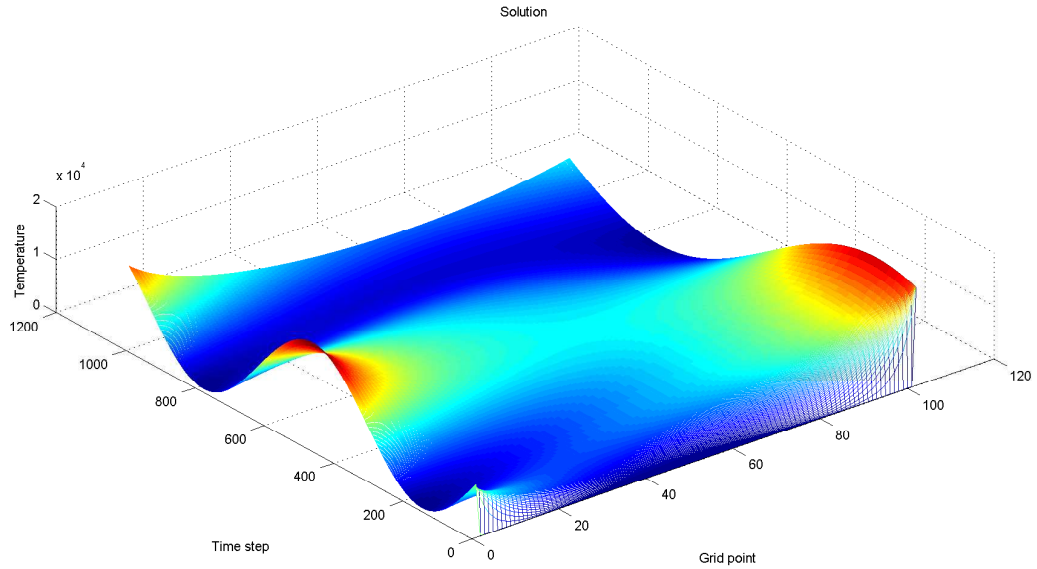


Figure 4: *Solution of the heating of a 1-D rod, with sine functions as boundary conditions. This solution (these boundary conditions) are used in the comparison of algorithm speed between methods II and III.*

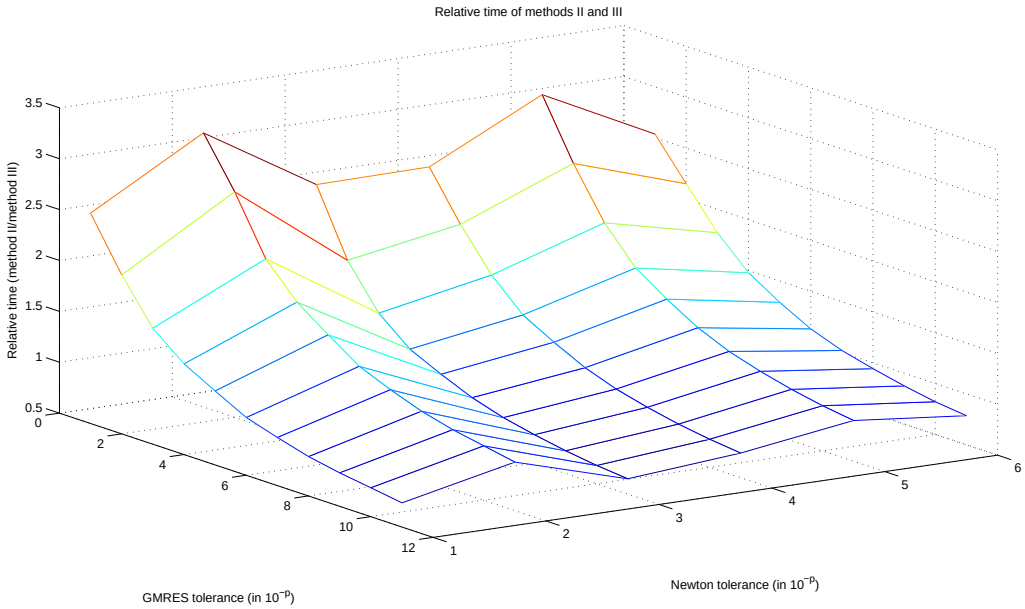
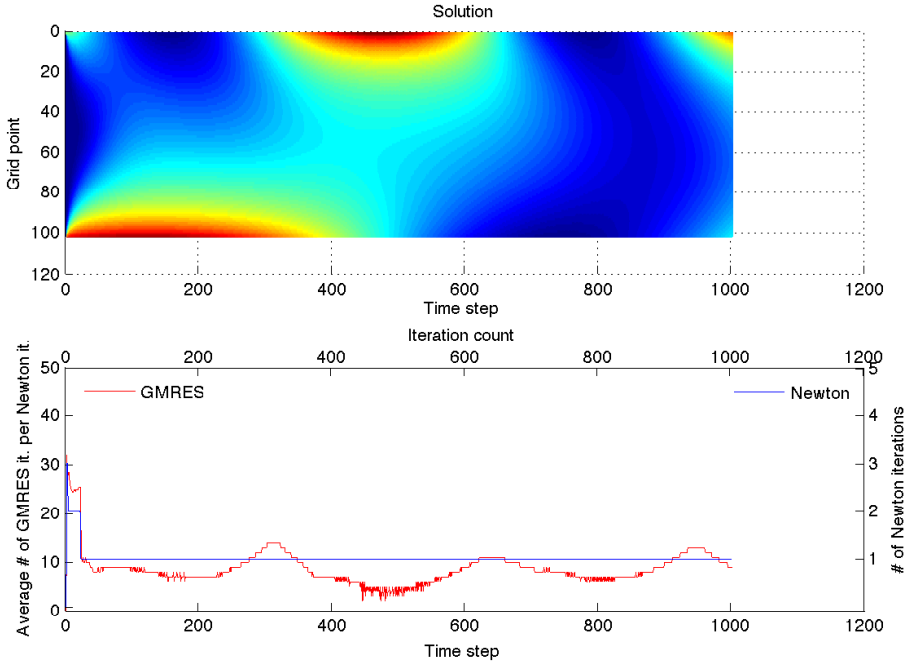
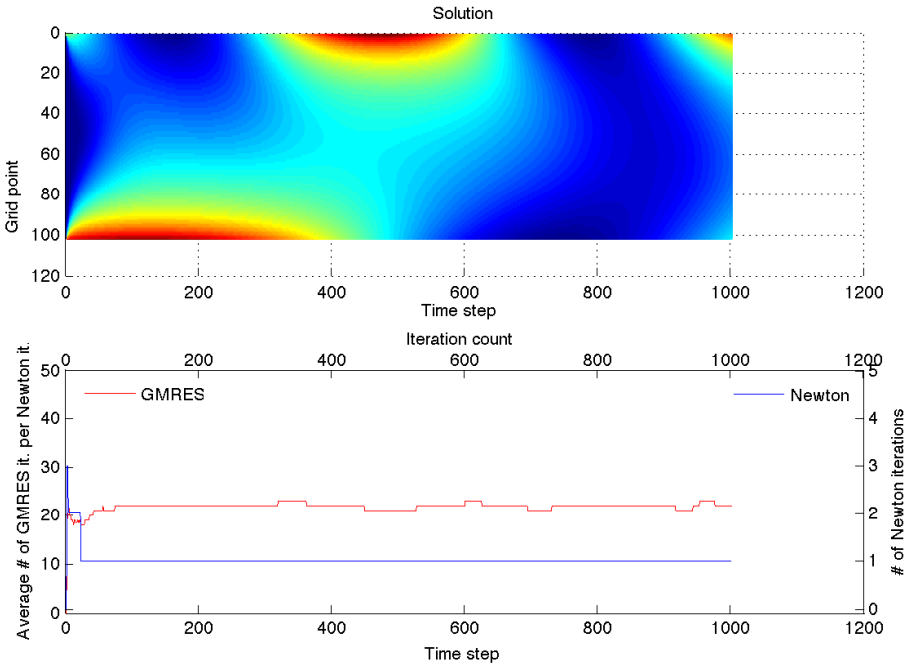


Figure 5: *Comparison of speed of methods II and III. Horizontal axes show the parameter p in $\tau = 10^{-p}$, where τ is the relative tolerance of either the Newton loop or the GMRES loop. Vertical axis shows the relative time, time of method II is divided by time of method III.*



(a) *Solution and number of iterations of method II. Newton iterations in blue and average GMRES iterations per Newton iteration in red.*



(b) *Solution and number of iterations of method III. Newton iterations in blue and average GMRES iterations per Newton iteration in red.*

Figure 6: *Comparison of the performance of the methods II and III with the problem of heating a 1-D rod with sine functions as boundary conditions.*

First of all it should be noted that the number of Newton iterations is the same in methods II and III. This is what one expects, because GMRES should converge to the same solution of the step that needs to be taken in the Newton iteration. However GMRES does not have the same amount of average iterations per Newton iteration in the two methods. This is due to the difference of the exact Jacobian matrix (method II) versus the approximation of the Jacobian matrix (method III). As expected the average amount of GMRES iterations per Newton iteration is higher in method III than in method II, because method III works with the approximation. Also the average number of GMRES iterations per Newton iteration is roughly constant in method III, but not in method II. The differences in method II arise because the problem is not always of the same difficulty. The number of average GMRES iterations per Newton iteration is constant in method III, because this number does not depend on the difficulty of the problem, but on the precision of the approximation of the Jacobian matrix.

3.4 Code coupling using method III

Three different ways of coupling are tested and compared with each other in this report, which are presented below. The coupled problem in this test is somewhat articial. The same rod is taken, but it is sliced in half. These two halves can only be calculated seperate from each other. The Jacobian matrix of this problem is known, for it is the same matrix as used in methods I and II of the previous section, but it is not used in this test. For clarity: these three methods are not the same as the three methods described before. Each of the coupling methods ((i), (ii) and (iii)) uses method III (JFNK), but the way the coupling happens differs. More on different ways of coupling problems can be found in [8].

3.4.1 Coupling method (i)

First of all the straight forward coupling method of solving each part seperately. Each part is solved independent of the other, using only the temperatures at the boundary of the other half. This program first calculates the left side of the rod (starting at $x = 0$) using the boundary condition on the left and the tempeture of the previous moment in time on the other side. Once the approximation of this half has converged to the desired error, the other half will be approximated. This half uses the boundary condition on the right and the temperature (of the current moment in time) of the other half on the left. This coupling is not always accurate, since there is no real measure of the error of the solution. Figure 7 shows a scheme of this coupling method, the dashed lines show a way of improving this method. If the second half is converged, one can go back to the first half and converge it again, using the value at the boundary with the right side (of the current moment in time). This ”spiral” of converging the halves can be continued untill some measure for the total error has decreased to the desired precision, then the program can continue to the next time step.

3.4.2 Coupling method (ii)

Coupling method (i) exchanges values at the boundary (half way up the rod) whenever a Newton loop has converged. The second coupling method differs in that it exchanges the values at the boundary whenever a GMRES loop has converged. Figure 8 shows a schematic of coupling method (ii). Left and Right should be interpreted as the GMRES loop of that respective side. There are no dashed lines in this schematic, since they were not used in the program of this research.

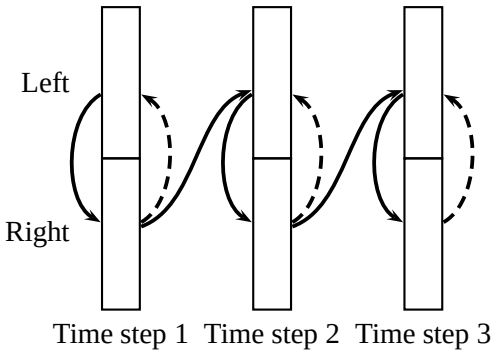


Figure 7: *Schematic of the simplest coupling method (coupling method (i)), solving each half of the rod seperately, before going to the next time step. The dashed lines are optional (one gets better accuracy following these).*

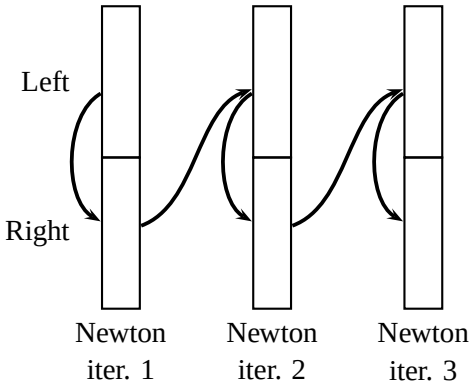


Figure 8: *Schematic of coupling method (ii). Now not the Newton loop of the halves is converged before continuing, but the GMRES loop is converged before going to the next Newton iteration.*

3.4.3 Coupling method (iii)

Coupling method (iii) uses the real strength of JFNK. Since the Jacobian matrix is not needed, just the function evaluations of the heat equation at all points in the rod are needed. The function **F** can be made by concatenating the evaluations of the heat equations of the two parts. Exactly like the example that is used in section 2.3.1. This means the exchange of values happens when the function **F** is evaluated, therefore the Jacobian matrix can be approximated. Coupling methods (i) and (ii) do not use an approximated Jacobian matrix or the real Jacobian matrix, while coupling method (iii) does. Coupling method (iii) has a real measure of the error in the solution, while coupling methods (i) and (ii) do not.

3.4.4 Comparison of the coupling methods

Since the rod is sliced in half, the temperature in the middle should vary to test this program. As this is not seen in the solution of the problem as in Figure 4, other boundary conditions are used, the solution of this problem can be seen in Figure 9. The initial condition is still zero Kelvin in the whole rod. The boundary conditions are

$$T_{left}(n) = T\sin(25\frac{\Delta t}{t}n + 3) + T \quad (30)$$

$$T_{right}(n) = T\sin(10\frac{\Delta t}{t}n + 1) + T \quad (31)$$

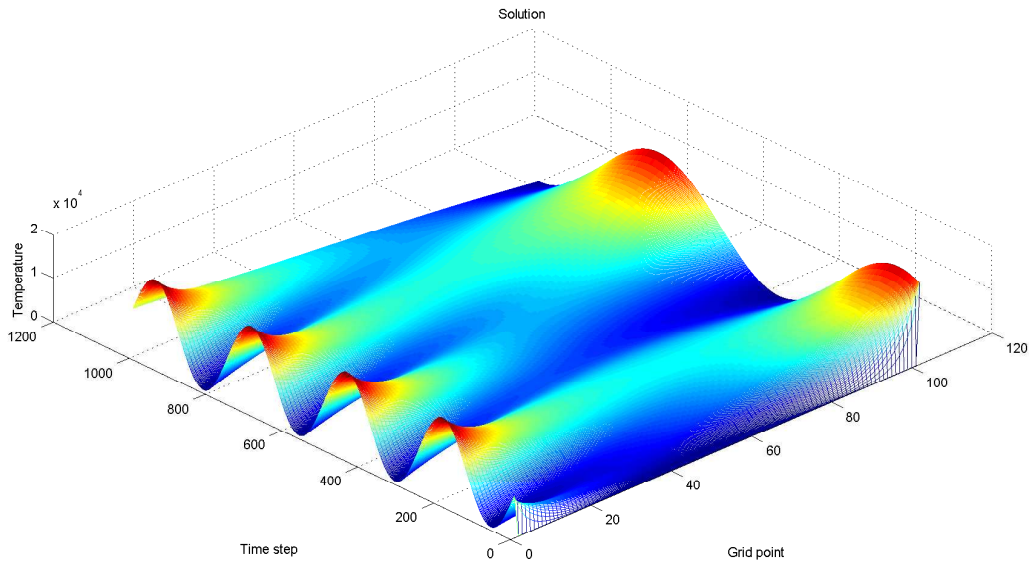


Figure 9: *Solution of the problem used for the tests of the code coupling programs. Another solutions is chosen because in this solution the temperature in the middle of the rod (where it is coupled) changes.*

Three comparisons are made in this section. First the Newton iteration counts and average GMRES iterations per Newton iteration counts of all three coupling methods are compared. Second the difference in time between the standard coupling method (i) and the JFNK coupling method (iii) is looked at. Third is a comparison of the accuracy of all three coupling methods when a large time step is used.

For the first test three iteration count plots (one for each coupling method) are presented and discussed. A plot of the solution with iteration count plots of coupling method (i) can be found in Figure 10. In this figure each half of the rod is converged only once, that means that the dashed lines in Figure 7 are not used. The plot in the middle is the iteration count of the right half (from $x = 51$ to $x = 100$), the plot on the bottom is the iteration count of the left half (from $x = 1$ to $x = 50$). It is easy to see the times at which the problem is easier or more diffucult. The Newton iteration tells us about this difficulty. When the temperature does not change in one time step, the Newton iteration does not have to take many steps and therefore finishes quicker. This is exactly what happens at the minima of the temperature at the boundaries. The minima of the boundary coincide with less Newton iterations in the half of that boundary. So because the left boundary has four minima, there are four times where less Newton iterations are used, these times coincide. This hold for the right half as well, only with two minima.

Figure 11 shows a plot of the solution and iteration count of coupling method (ii). In this plot the information of where the problem is hard or easy to solve is lost. This is because one does not know how fast the two GMRES loops combined converge, in other words with what error the program goes to the next Newton iteration (see Figure 8). The program only converges both GMRES loops once, it

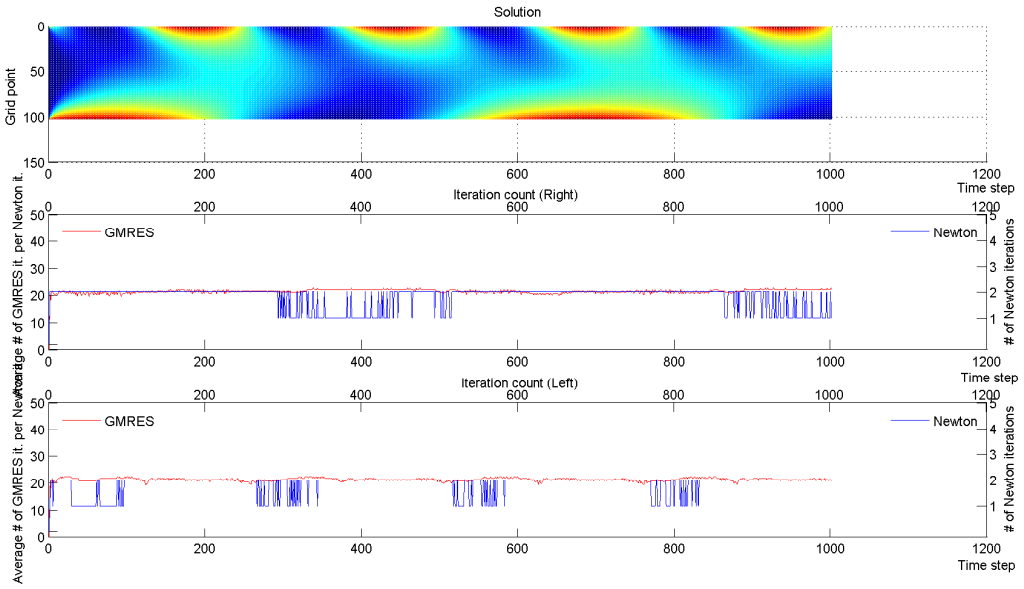


Figure 10: *Solution and number of iterations of coupling method (i) (the standard method). Newton iterations in blue and average GMRES iterations per Newton iterations in red. Tolerances are* 10^{-3}*. Upper plot is solution, middle plot is right half, lower plot is left half.*

is not known whether this is always good enough to stop the Newton iteration, therefore the number of Newton iterations is also affected by this. The number of Newton iterations will not provide a measure for how hard or easy the problems is either. Overall this method should not be used, since it provides less information about the difficulty of the problem at hand. Also it does not solve the problem faster than coupling methods (i) and (iii).

Coupling method (iii) uses the strength of JFNK, as explained earlier. An iteration count plot of coupling method (iii) is found in Figure 12. The amount of iterations, both Newton and GMRES, is roughly constant. This is not surprising because the JFNK method without coupling (method III) also had roughly constant iteration counts. This should be similar because only the evaluation of the function $\mathbf{F}$ is used to solve the problem, which is in both cases the same function (although different boundary conditions were used).

The reason why coupling method (iii) is better than coupling method (i) is seen in the second comparison of the methods. A plot of the relative speed of the methods is given in Figure 13. A value above 1 means the JFNK coupling was quicker, a value below 1 means the time level coupling was quicker. As can be seen in the plot, JFNK was quicker with all tested accuracies. For example if the tolerance for both the Newton iteration and the GMRES iteration is 10^{-3}, coupling method (iii) is almost three times faster. This means the true JFNK method (coupling method (iii)) of solving coupled problems is at least three times faster than standard coupling methods (coupling method (i), not with method III, but method I).

The last test of how the coupling methods perform, was to test in more difficult circumstances. This was done by increasing the time step, Δt. This makes the problem more diffucult because the Newton loop has to adjust more to find the solution. Figures 14, 15 and 16 show the results of this test. All three coupling methods were tested here.

In Figures 14 and 15 it is clearly seen the solution is not the right solution. These are the plots of

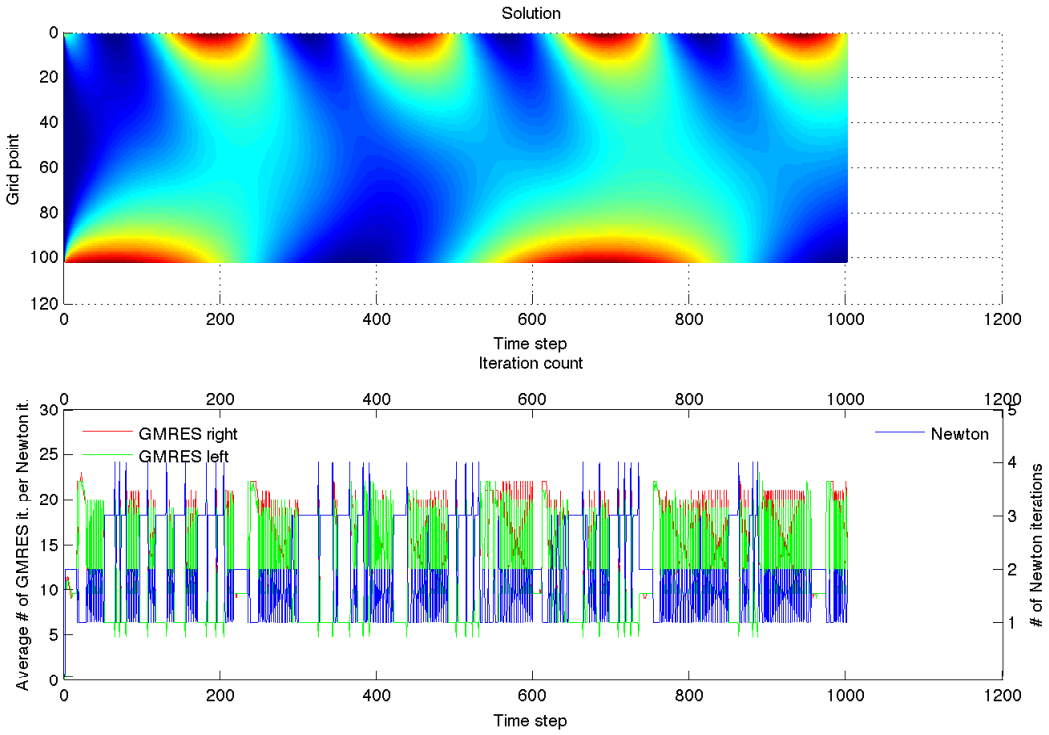


Figure 11: *Solution and iteration count plot of coupling method (ii). Tolerances are* 10^{-3}*. Upper plot is the solution, lower plot has Newton iterations (in blue), average number of GMRES iterations per Newton iteration of the left half (in green) and of the right half (in red).*

coupling methods (i) and (ii). At the boundary between the two coupled halves of the rod the temperature is in a local maximum or not smooth. Clearly these methods do not always work, since one does not know what the error is and where the error is. The location(s) of the error are the variables which contribute most to a norm used to define the error. For now the difference between solution of the coupling method and the real solution is used as error. Here all the error is concentrated along the boundary between the two halves.

However, the JFNK coupling method (coupling method (iii)) works fine under these circumstances, as can be seen in Figure 16. There is no strange local maximum in the middle of the rod, nor any irregularities. What does catch attention is the difference in iterations between methods (i) and (ii) on the one hand and method (iii) on the other hand. In the case of the JFNK coupling method (method (iii)) more iterations were needed. This is probably because the iteration count plots in methods (i) and (ii) contain only the count of the first time a loop converged. This means the first time the Newton loop converged in method (i) (Figure 14) and the first time both the GMRES loops converged in method (ii) (Figure 15). However method (iii) (Figure 16) converges only once, because then the final answer is obtained. Therefore the methods (i) and (ii) need more iterations than can be seen in these plots.

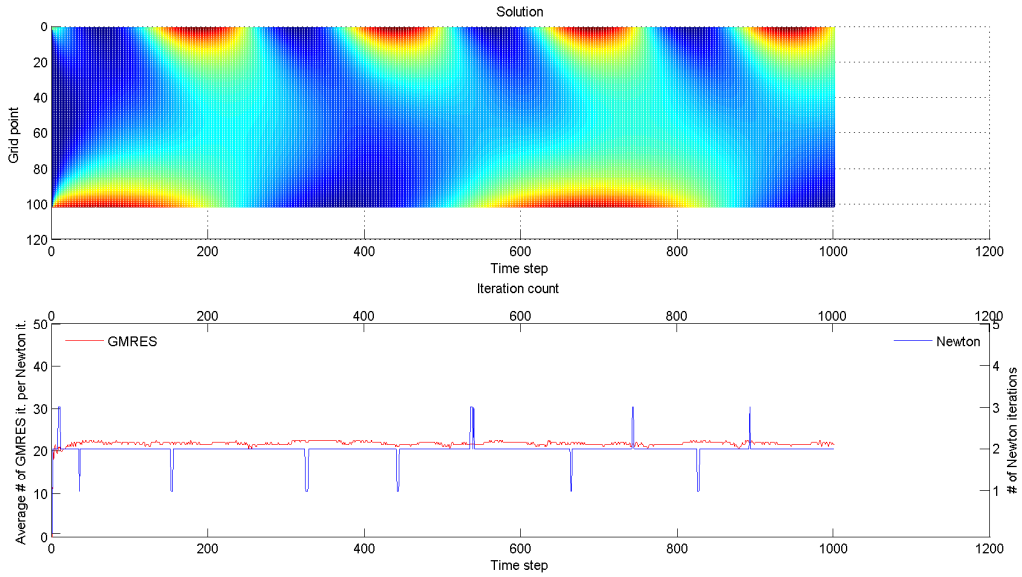


Figure 12: *Solution and number of iterations of coupling method (iii). Tolerances are* 10^{-3}*. Upper plot shows the solution, lower plot shows the iteration count plot. Number of Newton iterations in blue, average number of GMRES iterations per Newton iteration in red.*

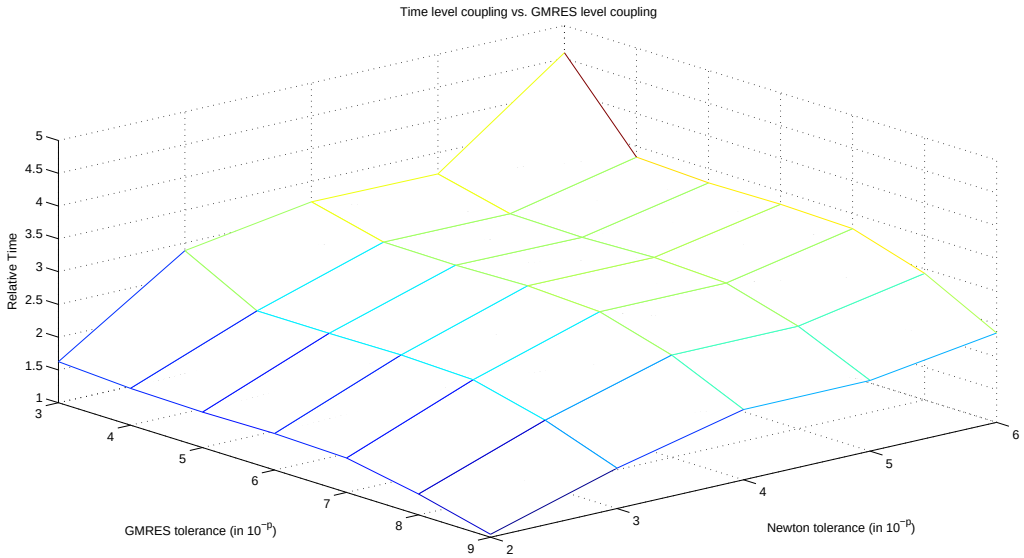


Figure 13: *Comparison of the standard coupling method (coupling method (i)) with the JFNK coupling method (coupling method (iii)). Relative time is the time of JFNK divided by the time of the standard method. The horizontal axes show the parameter* p *in* $\tau = 10^{-p}$*, where* τ *is the tolerance.*

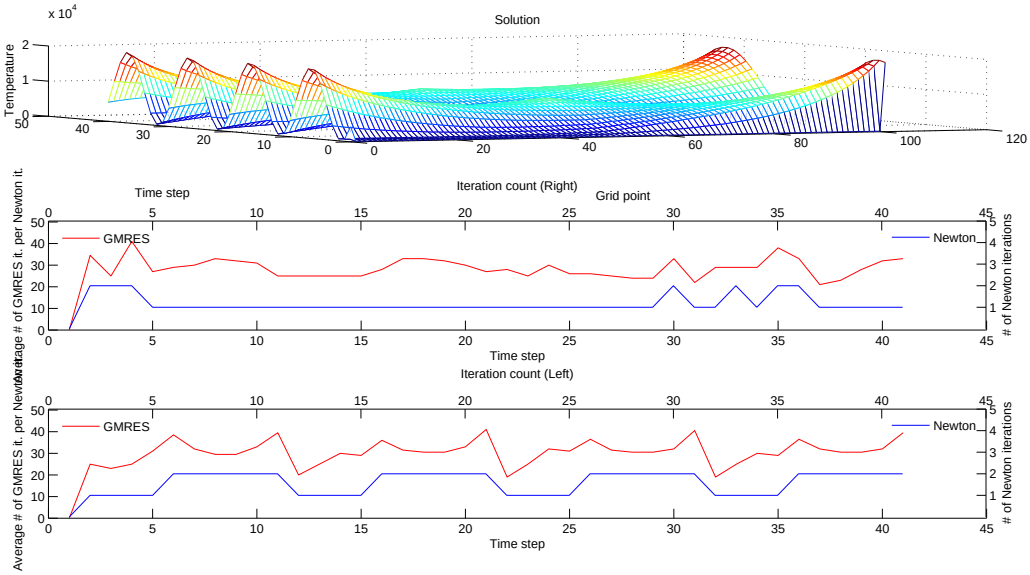


Figure 14: *Time level coupling. Solution of the problem with large Δt. This results in errors in the coupling area (in the middle of the rod) in ordinary coupling (coupling methods (i) and (ii)), not in the JFNK coupling method (coupling method (iii)). Upper plot is solution, middle plot is the iteration count plot of the right half, the lower plot is that of the left half. Number of Newton iterations is in blue, number of average GMRES iterations per Newton iteration isin red.*

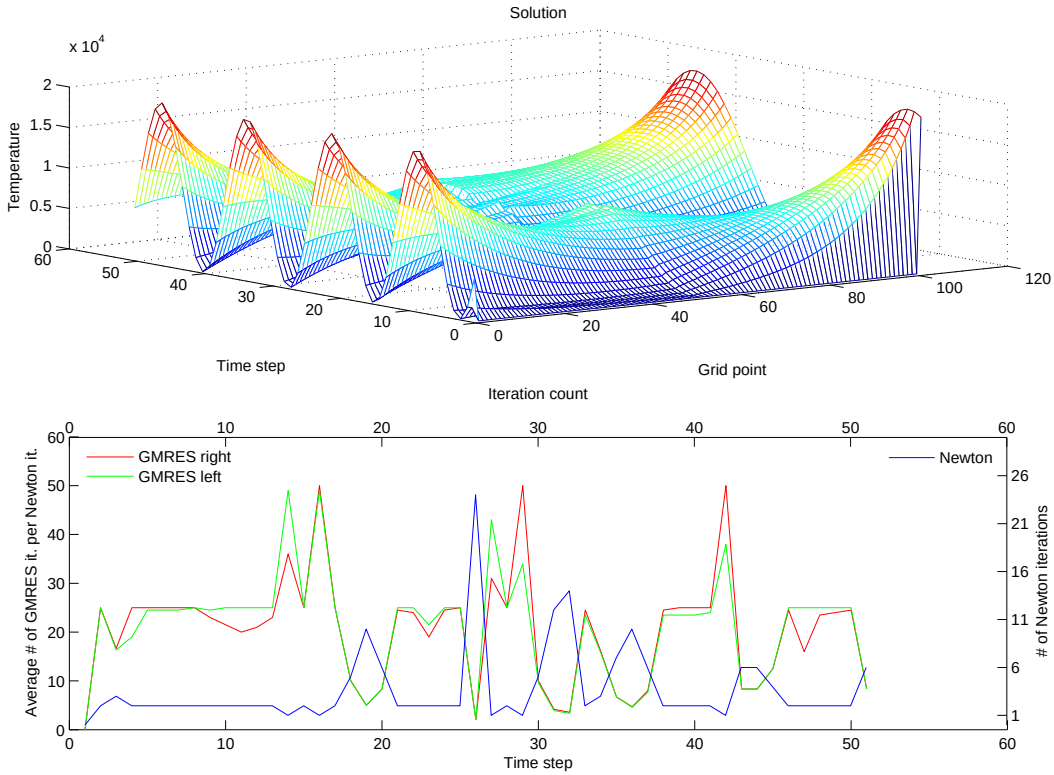


Figure 15: *Newton level coupling. Solution of the problem with large Δt. This results in errors in the coupling area (in the middle of the rod) in ordinary coupling (coupling methods (i) and (ii)), not in the JFNK coupling method (coupling method (iii)). Upper plot is solution, lower plot is the iteration count plot. The number of Newton iterations is in blue, the number of average GMRES iterations per Newton iteration of the left half is in green, that of the right half in red.*

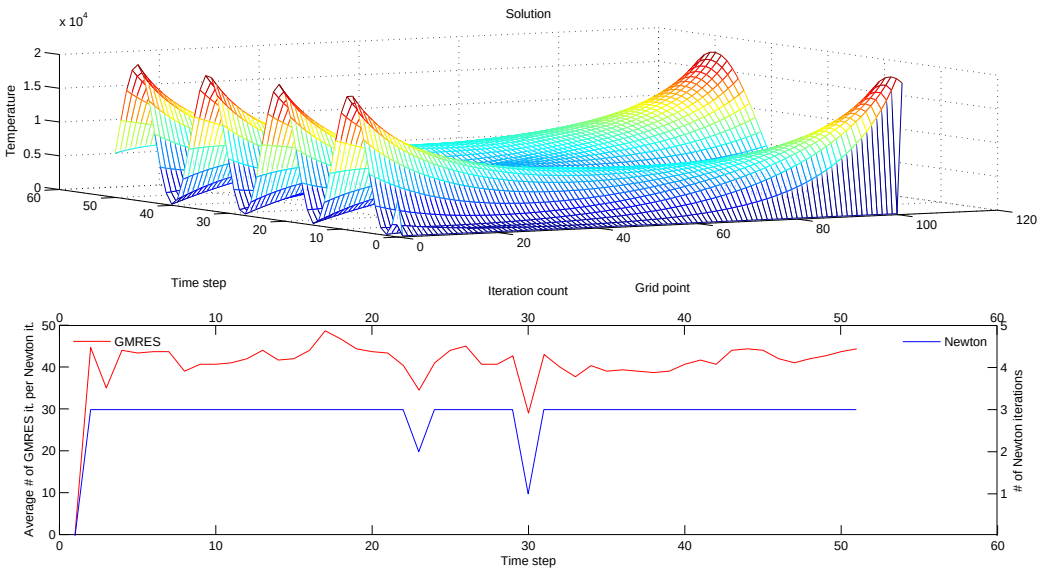


Figure 16: *JFNK coupling. Solution of the problem with large Δt. This results in errors in the coupling area (in the middle of the rod) in ordinary coupling (coupling methods (i) and (ii)), not in the JFNK coupling method (coupling method (iii)). Upper plot is solution, lower plot is the iteration count plot. Number of Newton iterations is in blue, the average number of GMRES iterations per Newton iteration is in red.*

4 Test problem 2: Molten salt reactor

A molten salt reactor (MSR) is a reactor in which the fuel is in solution with a molten salt. This allows for the fuel to be pumped around in a system of pipes and other components. The basic scheme of an MSR is found in Figure 17. In the first section a description of both the continuous time model and a discretization of that model will be given. The discretized model is used to do all computations of the JFNK method with. This test problem was used because the mathematics and especially the coupling is more difficult than in the first test problem. More difficult means that more Newton and GMRES iterations are needed to reach the desired tolerance. Also this is a problem in the area of nuclear reactor physics, which is of our interest.

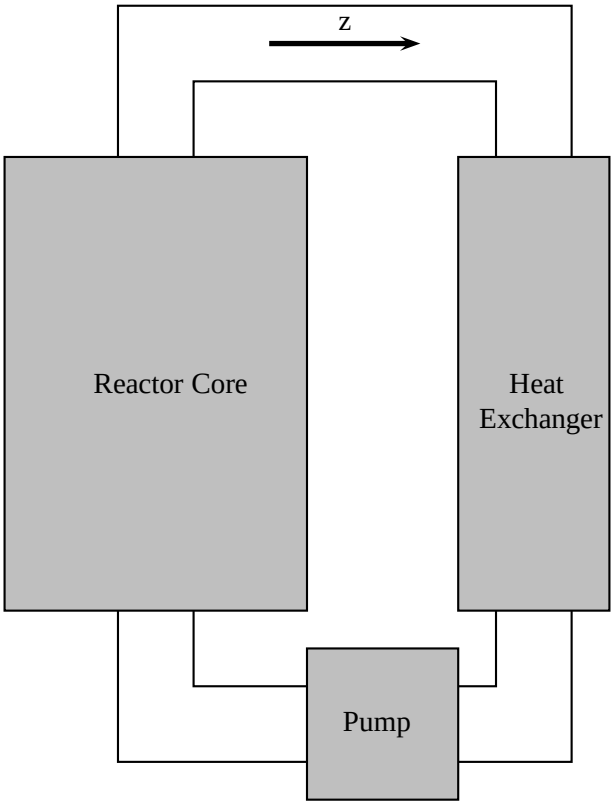


Figure 17: *Schematic view of a molten salt reactor, used in this project to test the JFNK method. An MSR consists of three elements with pipes in between to pump the salt around through the elements. The elements are the reactor core, where fission takes place. The heat exchanger, where the heat used to generate electricity is taken out of the reactor. The third is the pump, to circulate the fuel-salt mixture through the reactor.*

4.1 Mathematical and computational model

4.1.1 Geometry of the MSR

There are three essential components of an MSR: the reactor core, the heat exchanger and the pump. The reactor core is - in this model - a large cilindrical shaped vessel. This is where fission of the uranium fuel

takes place and thus precursors are produced. In a reactor where the fuel is not moving the precursors decay on the same spot as where they were produced. However, in this model the fuel is being pumped around and therefore the decay of the precursors takes place elsewhere, even outside the reactor core. The core is not just a hollow vessel, the fuel in the molten salt flows past bars of graphite, in order to control the fission process.

The heat exchanger is used to extract heat from the fuel salt mixture and can be used to generate electricity or for other purposes. The pump is used to transport the fuel salt mixture around the system, without a pump the mixture in the core cannot cool down and the fuel elsewhere in the system is not used in the fission process.

The model takes only one spatial dimension into account, differences in the radial direction are neglected. The parameter z in Figure 17 denotes the direction of this spatial parameter. The positive z-direction is also the direction of the flow in this model, the flow can however be easily reversed. The parameter z is discretized in an integer number of elements, with different cross sectional areas in the different components of the MSR.

4.1.2 Neutronics model

The neutronics model used is an adaptation of the standard point kinetics model. The model will consider only the total amount of neutrons present in the reactor core in each time step. The model differs from the standard point kinetics model for the amount of precursors, since the precursors are being pumped around. Therefore the model describes the density of precursors in each element. The equations used in the model are derived from the "standard" point kinetics model, which are

$$\frac{dN(t)}{dt} = \frac{\rho(t) - \beta}{\Lambda} N(t) + \lambda C(t) \tag{32}$$

$$\frac{dC(t)}{dt} = \frac{\beta}{\Lambda} N(t) - \lambda C(t) \tag{33}$$

Wherein N is the total amount of neutrons in the reactor core, C is the total amount of precursors present in the core. The equations are differential equations in these two quantities. $\rho(t)$ is the reactivity of the core at time t, β is the delayed neutron fraction. λ is the precursor decay time, Λ is the neutron generation time. In this model only one precursor group is taken into account, if one wants to look at the effect of several precursors the term $\lambda C(t)$ changes into a sum over all different precursors and for every different precursor group a seperate equation like equation 33 has to be used. The derived equations become

$$\frac{\partial N(t)}{\partial t} = \frac{\rho(T) + \rho_{ext}(t) - \beta}{\Lambda} N(t) + \lambda \frac{\int_0^H A(z) C(z,t) \phi(z) dz}{\int_0^H f(z) \phi(z) dz} \tag{34}$$

$$\frac{\partial C(z,t)}{\partial t} = \frac{\beta f(z)}{\Lambda A(z)} N(t) - \lambda C(z,t) - \frac{\partial}{\partial z} \frac{g(z,t) C(z,t)}{A(z)} \tag{35}$$

wherein N still denotes the total amount of neutrons, $C(z)$ however denotes the precursor density at position z. This means some changes in the form of the equations, although they still mean the same. The changes involve the cross sectional area $A(z)$ at z and the fission shape $f(z)$. The fission shape, in brief, determines where fission in the geometry of the reactor takes place. In this model f is taken as $f(z) = \frac{1}{H}$ in the reactor core, with H the height of the reactor core, which is in this case the length of the reactor core along the z-axis. Outside the reactor core f is taken as $f(z) = 0$. $\phi(z)$ is the adjoint-flux, which is to describe the relative probability that a neutron created at a certain location will contribute to

the fission process. It is taken to be 1 inside the core and 0 outside the core. In the differential equation of C the flow of the fuel salt mixture and therefore the precursors is also taken into account, with $g(z,t)$ being the flow rate at position z at time t. Also a difference is made between the temperature feedback on the reactivity and the external reactivity. Temperature feedback will be discussed later on, the external reactivity is a parameter that simulates external sources of radiation or extra graphite near the reactor core.

Because the total number of precursors is changed into a density of precursors, one has to integrate equation 35 over the whole reactor to get equation 33 back. If one then multiplies the equation with a factor of $A(z)$, the original point kinetics equation is obtained. This means the terms in equation 35 have a different dimension than equation 33.

Equations 34 and 35 are discretized before they can be used in JFNK and become

$$\frac{N^{n+1}-N^n}{\Delta t} = \frac{\rho^{n+1}+\rho_{ext}^{n+1}-\beta}{\Lambda}N^{n+1} + \lambda\frac{\sum_j A_j \Delta z C_j^{n+1}\phi_j}{\sum_j f_j \phi_j} \tag{36}$$

$$\frac{C_j^{n+1}-C_j^n}{\Delta t} = \frac{\beta f_j N^{n+1}}{\Lambda A_j \Delta z} - \lambda C_j^{n+1} - \frac{g^{n+1}C_j^{n+1}}{A_j \Delta z} + \frac{g^{n+1}C_{j-1}^{n+1}}{A_j \Delta z} \tag{37}$$

In these equations some extra notation is introduced. Δz denotes the size of an element in the z-direction, Δt is the time discretization. The subscript j on any variable means it is the value of element j of that variable and the superscript n or $n+1$ denotes the value of that variable on the current or the next instant in time. Here g is taken to be constant throughout the reactor and in time. The flow rate has to be a constant throughout the reactor, or fuel-salt mixture will accumulate at one point in the reactor. It is also taken to be constant in time as to simplify the outcomes of the model. An important aspect of the discretized equations is that this set of equations only holds when $g(z,t) \geq 0$ (upwind model), because of the way the derivative with respect to z is discretized. If it is discretized otherwise, the flow rate can be negative and zero (downwind model).

4.1.3 Thermo hydraulic model

The thermo hydraulic model used for this simulation takes three effects into account. The production of heat by fission of the fuel in the fuel-salt micture, heat transport by the forced convection of the pump in the reactor cycle and the cooling of the fuel-salt mixture in the heat exchanger. Each of the three modeled effects are discussed seperately below.

The equation

$$\dot{q}'''(z,t) = \frac{p_{fiss} f(z)}{\Lambda \nu A(z)} N(t) \tag{38}$$

is used to describe production of heat in the model, it is derived from the expression of the total generated heat in the reactor core, which is given by $P = N v \Sigma_F p_{fiss}$. In this expression P stands for the total heat generated in Watts, N is the number of neutrons inside the core, v is the average velocity of the neutrons in the core (in cm/s). Σ_F is the macroscopic fission cross section, which is the probability a neutron collides and induces fission per centimetre traveled inside the core. p_{fiss} is the average amount of heat a nucleus generates when a neutron induces a fission of the nucleus. Equation 38 is derived by multiplying the expression for the total amount of generated heat by $f(z)/A(z)$, which gives an expression of the heat generated per unit volume, which is denoted by $\dot{q}'''$. With the definition of $\Lambda = (v\nu\Sigma_F)^{-1}$ the derivation is complete, ν being the average number of neutrons produced per fission.

The change in energy at a certain point z due to the forced convection of the pump is given by $-\partial(uh_{fuel})/\partial z$. In this expression $u(z,t)$ is the speed of the fuel-salt mixture and $h_{fuel}(z)$ is the thermal energy of the fuel-salt mixture. This results in

$$\phi'''_{convection} = -(\rho c_p)_{fuel}\frac{\partial}{\partial z}\frac{g(z,t)}{A(z)}T(z,t) \tag{39}$$

when these two relations are used: $u(z,t) = g(z,t)/A(z)$ and $h_{fuel}(z) = (\rho c_p)_{fuel}T(z,t)$. In these relations ρ is the density of the fuel-salt mixture and c_p is the heat capacity of the fuel-salt mixture.

The heat flow out of (or into in rare cases) the fuel-salt mixture in the heat exchanger is modeled according to Newton's Law of Cooling, which is $\dot{\phi}_q = hA\Delta T$. $\dot{\phi}_q$ is the heat flow out of the fuel-salt mixture into the coolant, h is the heat exchange coefficient, which is a heat exchanger and flow-rate dependent parameter and ΔT is the temperature difference between the fuel-salt mixture and the coolant. The temperature of the coolant is chosen to be constant for simplicity. The expression for the heat flow per volumic unit, $\dot{\phi}'''$, is

$$\dot{\phi}'''(z) = \frac{h(z)O(z)}{A(z)}(T(z,t) - T_{he}) \tag{40}$$

in which T_{he} is the temperature of the coolant in the heat exchanger and $O(z)$ is the circumference of the heat exchanger, which defines the area the heat can flow through.

The three described effects provide terms for the right hand side of the equation, the left hand side consists of the change in heat in the fuel, that would be $\partial(h_{fuel})/\partial t$. For simplicity the temperature of the graphite in the reactor core is taken to be equal to the temperature of the fuel-salt mixture. This means that when the temperature of the fuel-salt mixture changes, the temperature of the graphite changes and therefore $\partial(h_{fuel})/\partial t$ does not cover the total change in energy. Instead $h_{total}(z) = h_{fuel} + h_{graphite}(z)$ is introduced, which is the total thermal energy, of both the fuel-salt mixture and the graphite in the reactor core. h_{total} can be written as $h_{total} = (\rho c_p)_{total}(z)T(z,t)$, with $(\rho c_p)_{total}$ as

$$(\rho c_p)_{total}(z) = \begin{cases} (\rho c_p)_{fuel} + c_{p,graphite}M_{graphite}/V_{fuelincore} & \text{Inside core} \\ (\rho c_p)_{fuel} & \text{Outside core} \end{cases} \tag{41}$$

Where $M_{graphite}$ is the total mass of graphite in the core and V_{fuel} is the total volume of fuel in the core. The values for $(\rho c_p)_fuel$ are 4 outside the core and 8 inside (in J/cm^3), in all models in this report. The total energy balance equation then becomes

$$(\rho c_p)_{total}(z)\frac{\partial}{\partial t}T(z,t) = \frac{p_{fiss}f(z)}{\Lambda\nu A(z)}N(t) - (\rho c_p)_{fuel}\frac{\partial}{\partial z}\frac{g(z,t)}{A(z)}T(z,t) - \frac{h(z)O(z)}{A(z)}(T(z,t) - T_{he}) \tag{42}$$

which takes the three effects into account and uses the correction for the changes in temperature of the graphite. The discretized equation is then

$$(\rho c_p)_{total,j}\frac{T_j^{n+1} - T_j^n}{\Delta t} = \frac{p_{fiss}f(z)}{\Lambda\nu A(z)}N^{n+1} - (\rho c_p)_{fuel}\left[\frac{gT_j^{n+1}}{A_j\Delta z} - \frac{gT_{j-1}^{n+1}}{A_j\Delta z}\right] - \frac{h_jO_j}{A_j}(T_j^{n+1} - T_{he}) \tag{43}$$

4.1.4 Temperature feedback

The only feedback onto the reactivity taken into account in this model is that of the temperature. The core is modelled as critical at a certain equilibrium temperature T_0, with a temperature differential as

$$\frac{\partial \rho}{\partial < T >} = \alpha \tag{44}$$

where $< T >$ is an average of the temperature in the reactor core [3]. Since the α is a constant, the reactivity is proportional to the average temperature. However proportionality leaves room for a constant between the average temperature and the external reactivity. This constant introduces the equilibrium temperature at which the reactor is critical. Which means the reactivity as a result of the temperature feedback (not the external reactivity) can be written as

$$\rho^{n+1} = \alpha(\sum_j f_j T_j^{n+1} - T_0) \tag{45}$$

The sum in this expression can be seen as a weighted average of the temperature, it is the average temperature of the reactor core. This equation is directly substituted in equation 36 in the model.

4.2 Three versions of the MSR

There are three versions of the model of the Molten Salt Reactor (MSR), in each version more effects are taken into account. This was done in order to build up the program in steps, as to find out where the JFNK method might not work. In the following three paragraphs the model of each version is described, as well as the results of the three versions. Also the encountered problems are mentioned and, when solved, their solution.

4.2.1 Version 1: Point kinetics

The standard point kinetics equations are solved using the JFNK method, in this version. The point kinetics equations are found in equations 34 and 35. In this version the fuel is stationary and there is no thermal feedback. The thermohydraulics part of the model is not considered either in this version. This version is used to compare the JFNK method to a direct solver, to see whether the JFNK method produces the right answers.

Three situations are used to compare the two methods with both small and large time steps. ρ in the point kinetics equations is the external reactivity, which means a positive value for ρ means a supercritical reactor, a negative value for ρ means the reactor is subcritical. The tests that are presented here are with $\rho = +\beta/2$, a supercritical reactor. This is considered on two time scales, a large scale with $\Delta t = 1\mathrm{s}$ and a small scale with $\Delta t = 1\mathrm{ms}$.

In the situation of a supercritical reactor core ($\rho = +\beta/2$) the number of neutrons and number of precursors should grow exponentially. There should also be an initial jump in the number of neutrons [4]. This jump should be of height $\beta/(\beta - \rho)$ times the initial value of the number of neutrons. The other two situations where $\rho = 0$ and $\rho < 0$ have also been tested, but are not presented here. In the case of a critical reactor ($\rho = 0$) the number of neutrons and precursors should remain constant. In the case of a subcritical reactor ($\rho < 0$) the number of neutrons and precursors should decrease, also with the initial jump but in the opposite direction (downward). The model performed well in these situations.

Figures 18 and 19 are the plots of the direct solver and of the JFNK method. As can be seen the two models produce the same solution. The initial conditions are $N = 100$ and $C = 50,000$. Also it is

not a difficult problem, since not many Newton and GMRES iterations are needed to solve the problem. However the JFNK algorithm needed two Newton steps in the case of $\Delta t = 1$ (Figure 18). This can be brought down to one iteration by making the tolerance for GMRES smaller.

4.2.2 Version 2: Feedback and spatial model

Version 2 uses the point kinetics equations with the spatial distribution of the precursors, as in equations 34 and 35. The temperature is accounted for in this version as well, however the reactor core cannot cool down, since the fuel-salt mixture is not being pumped around. Only the part of the fuel-salt mixture that happens to be in the heat exchanger can be cooled, the fuel-salt mixture that is in the reactor core will never lose its heat to the heat exchanger. This version also uses the temperature feedback.

A slight adjustment to the model is made, in order to test this version. Since the fuel-salt mixture in the reactor core cannot cool down, the reactor will always shut itself down, because of the temperature feedback. Therefore the heat exchanger is enlarged so the whole reactor system is cooled down, the reactor core included. When the core is cooled, the reactor does not shut itself down and tests with the external reactivity (ρ_{ext}) can be done.

Figure 20 shows a plot of the number of neutrons, the number of precursors, the temperature and iteration counts in the shut down situation, the reactor core is not cooled in this plot. The plot in Figure 21 shows the same variables, but now the reactor core is cooled, so it reaches its equilibrium. This equilibrium is verified by substituting all values of the equilibrium state in the governing equations, equations 34 , 35 and 42. In both plots the initial conditions are $N = 10^{11}$, $C = 10^8$ and $T = 600$. This problem is again an easy problem to solve for the JFNK method, since only one Newton iteration is used most of the time, only in the first few steps more than one iteration is used.

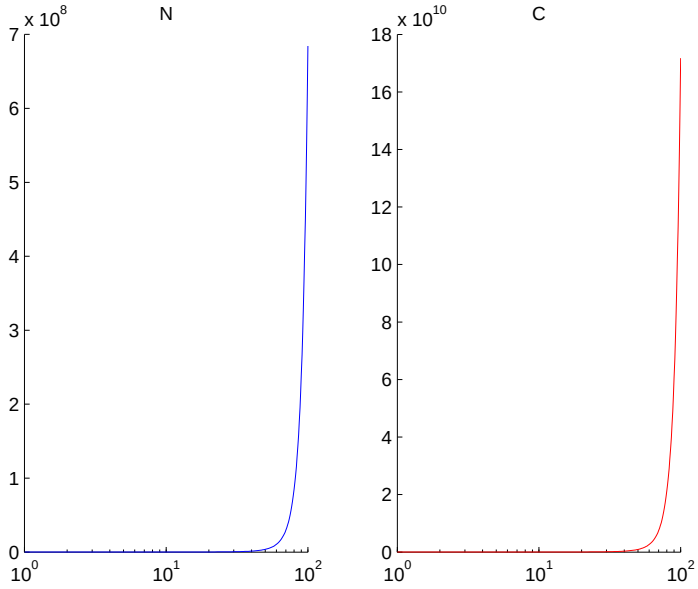
One of the inputs in the model is the external reactivity, which can be varied over time. To test how the model handles changes in the external reactivity, some rather odd functions were used as input for the external reactivity. The input functions are sine functions, jumps and ramps. Some input functions that the model handles well can be seen in Figures 22 and 23.

However, not all jumps are handled correctly by the model. Whenever a jump of about $+\beta$ is made in the input function of the external reactivity, the JFNK method does mostly not find the right solution. It does, in fact, find a solution of the equation, this is however not always a relevant solution. The set of governing equations does not have only physical realizable solutions, some are not realizable. The non-relevant solution is, in all cases investigated in this research, easily spottable. In such a case some quantities become negative at some point in the simulation. A plot of this situation is shown in Figure 24.

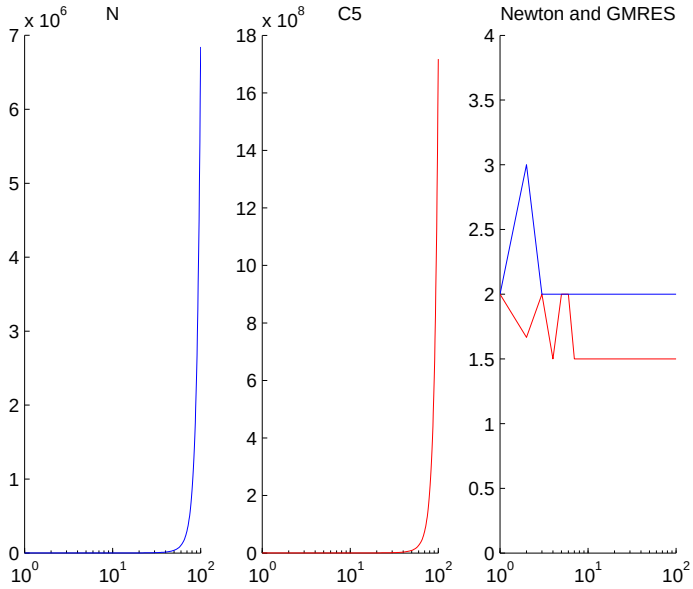
This peculiar behaviour can be corrected by changing the time step whenever such a step in the external reactivity occurs. The JFNK method will provide the correct solution when the time steps of the order of one milisecond are used, however, to simulate the same running time of the reactor the number of total steps will be several orders of magnitude larger. Since this is not practical, it is best to locally change the time step magnitude when the times at which the jumps occur are known or introducing variable timestepping in the algorithm. A Figure of the system with a jump in the external reactivity and small time step is shown in Figure 25.

Also notice that in all problems with a non-zero external reactivity more Newton iterations are needed to solve the problem. In general the number of iterations needed is between one and six. This means a problem with external reactivity is a harder problem than the problem in test problem 1.

Another problem occurs when the model is used with large differences in the numbers of neutrons, precursors and the temperature. This problem can be evaded by using relative variables, as discussed in the section mathematical theory. However, this does not guarantee a correct solution, even these relative

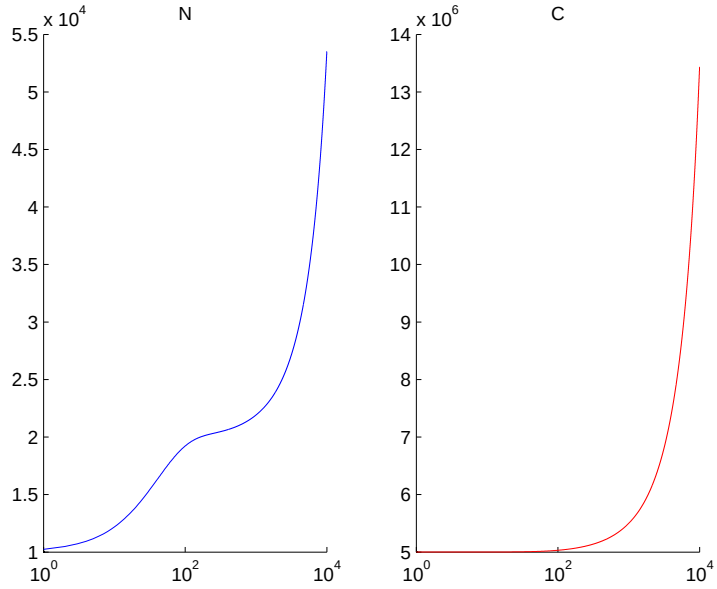


(a) *Solution with a direct solver. N is the number of neutrons, C is the number of precursors.*

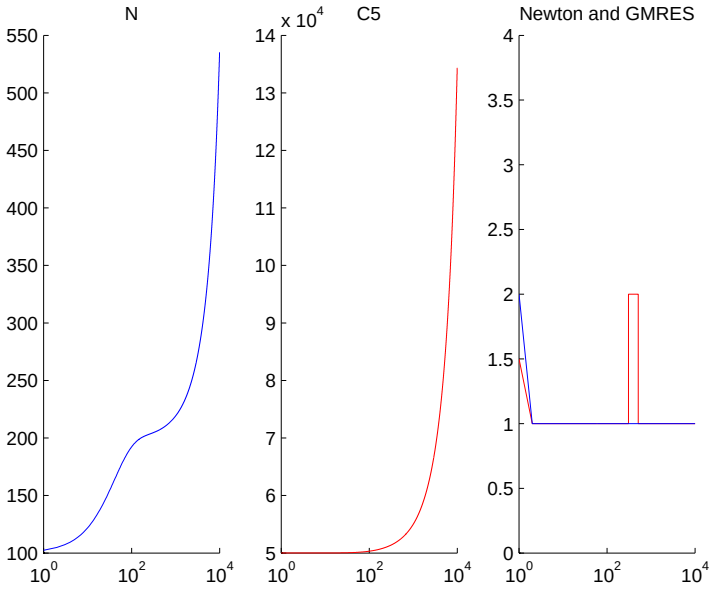


(b) *Solution with JFNK. N is the number of neutrons, C5 is the number of precursors. Iteration count plot is on the right, number of Newton iterations in blue, number of average GMRES iterations per Newton iteration in red.*

Figure 18: *Comparison of solutions produced by a Matlab solver and the JFNK method. In this example the reactivity is* $\rho = \frac{\beta}{2}$*, the time step is* $\Delta t = 1$.



(a) *Solution with a Matlab solver. N is the number of neutrons, C is the number of precursors.*



(b) *Solution with JFNK. N is the number of neutrons, C5 is the number of precursors. Iteration count plot is on the right, number of Newton iterations in blue, number of average GMRES iterations per Newton iteration in red.*

Figure 19: *Comparison of solutions produced by a Matlab solver and the JFNK method. In this example the reactivity is* $\rho = \frac{\beta}{2}$*, the time step is* $\Delta t = 10^{-3}$.

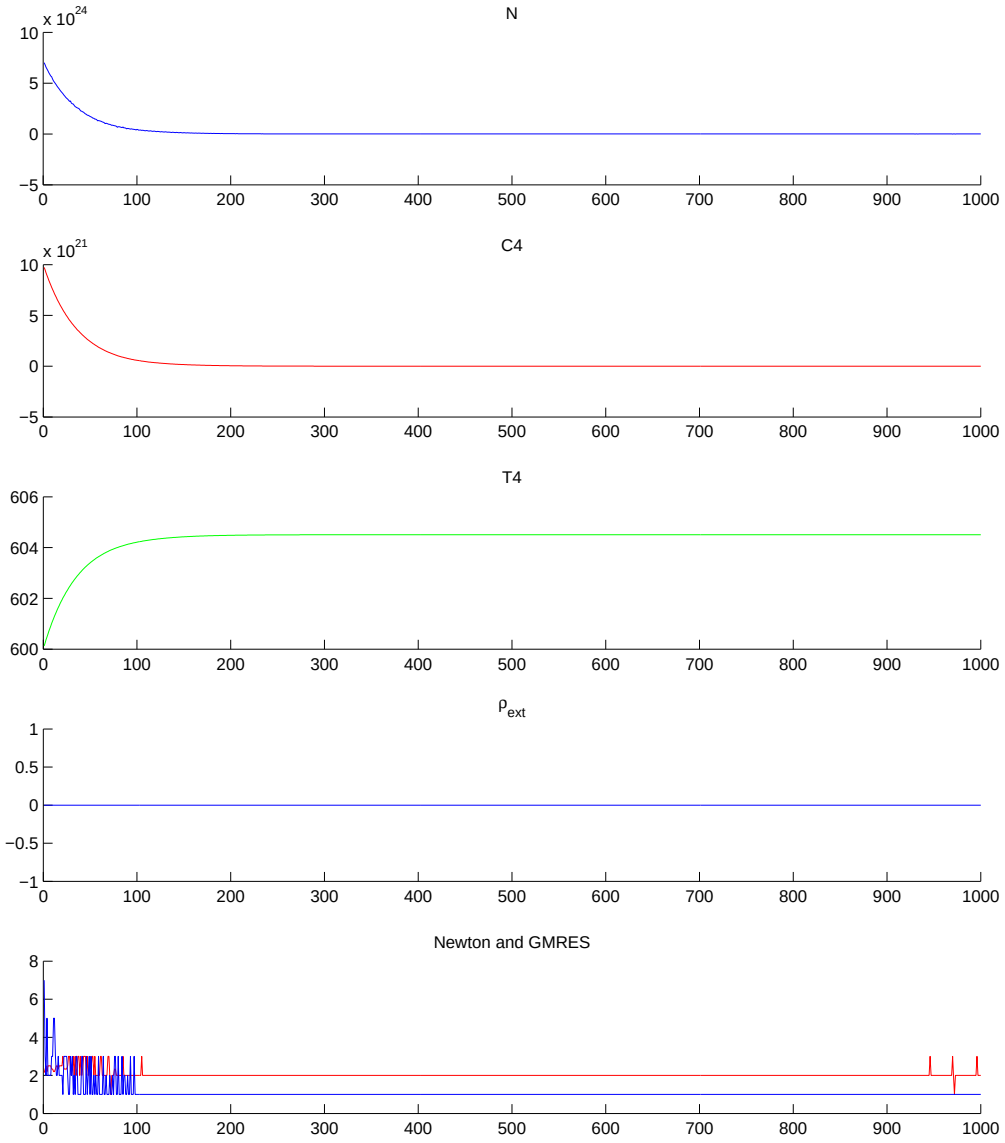


Figure 20: *Plot shows the number of neutrons in the reactor, number of precursors in cell 4 (which is in the core), the temperature of cell 4, the external reactivity and the number of iterations. The reactor core is not cooled, therefore the reactor shuts itself down.*

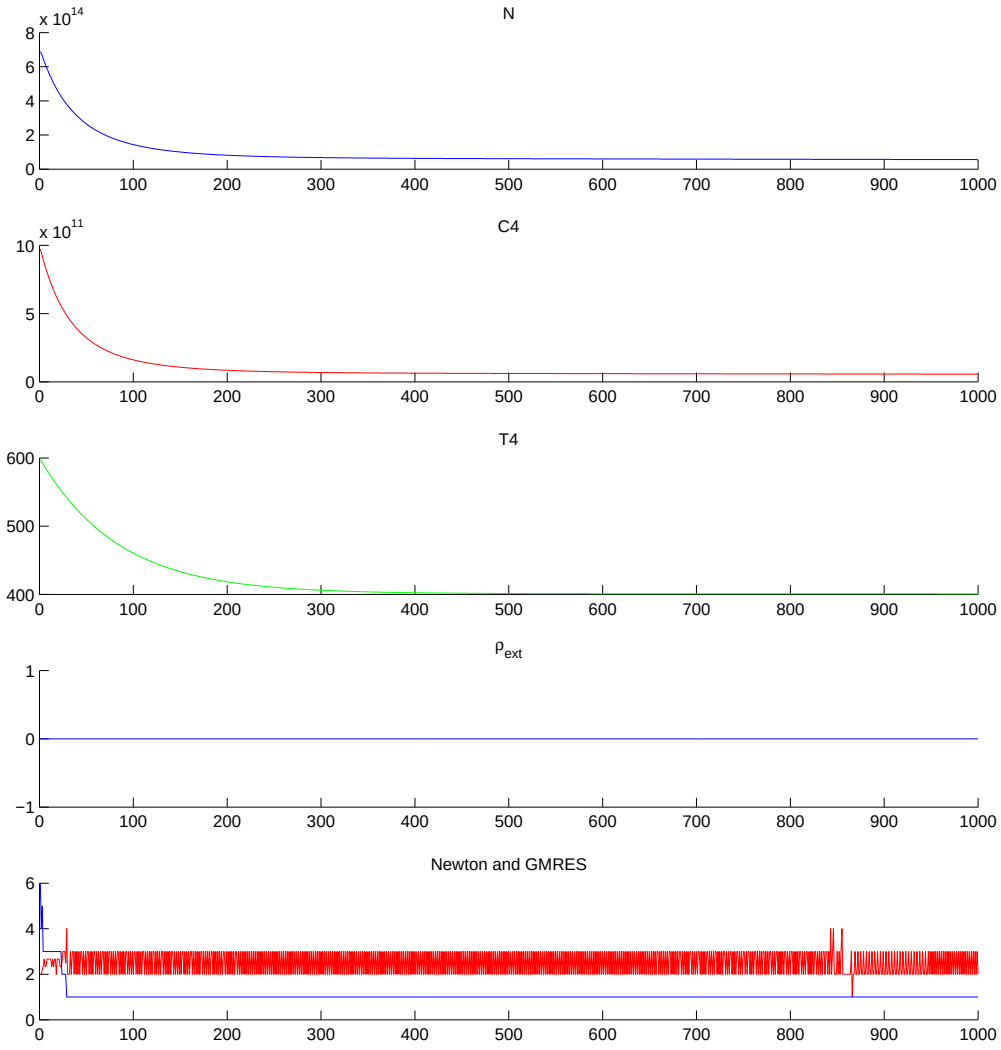


Figure 21: *Plot shows the number of neutrons in the reactor, number of precursors in cell 4 (which is in the core), the temperature of cell 4, the external reactivity and the number of iterations. The reactor core is cooled, therefore it reaches an equilibrium state.*

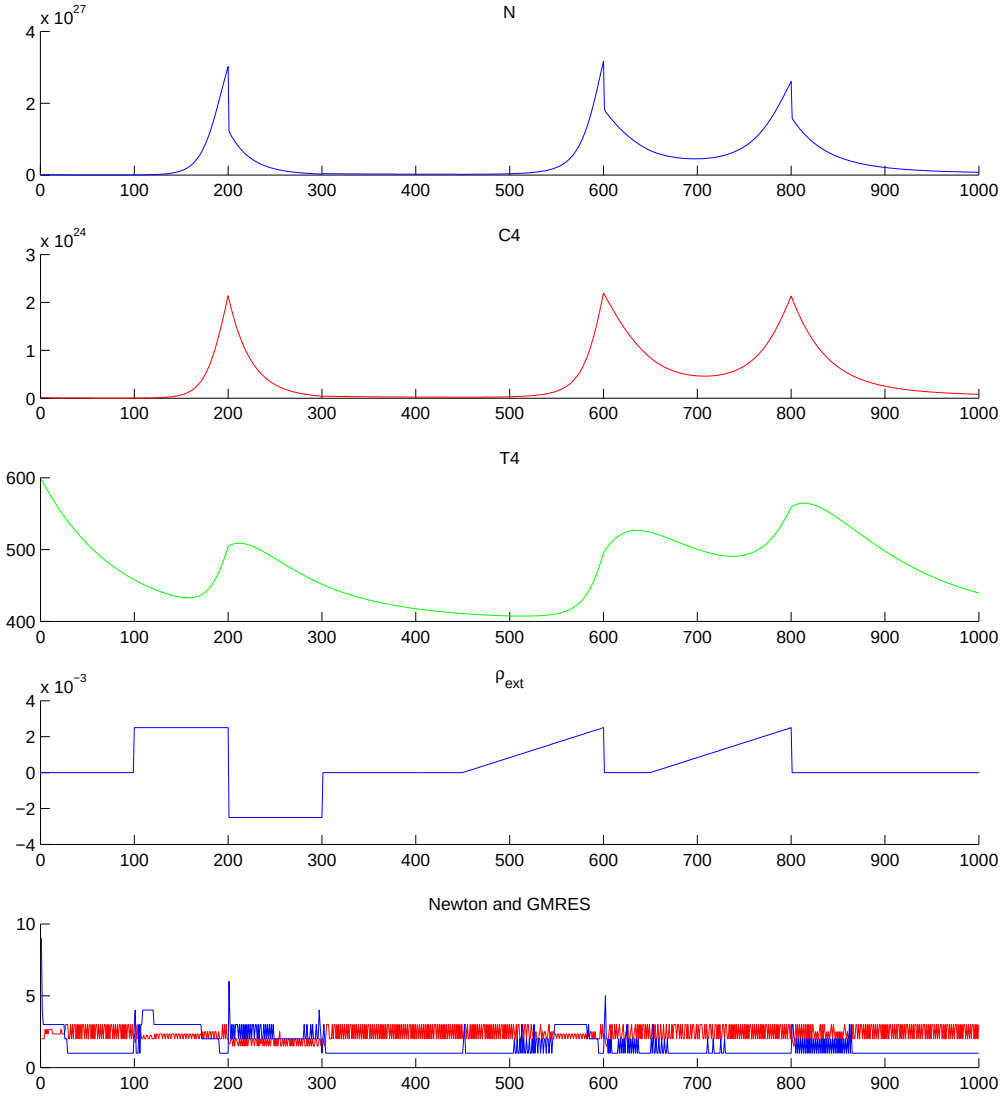


Figure 22: *As function for ρ_{ext} some jumps and two ramps are used to test the model. This function is handled with succes.*

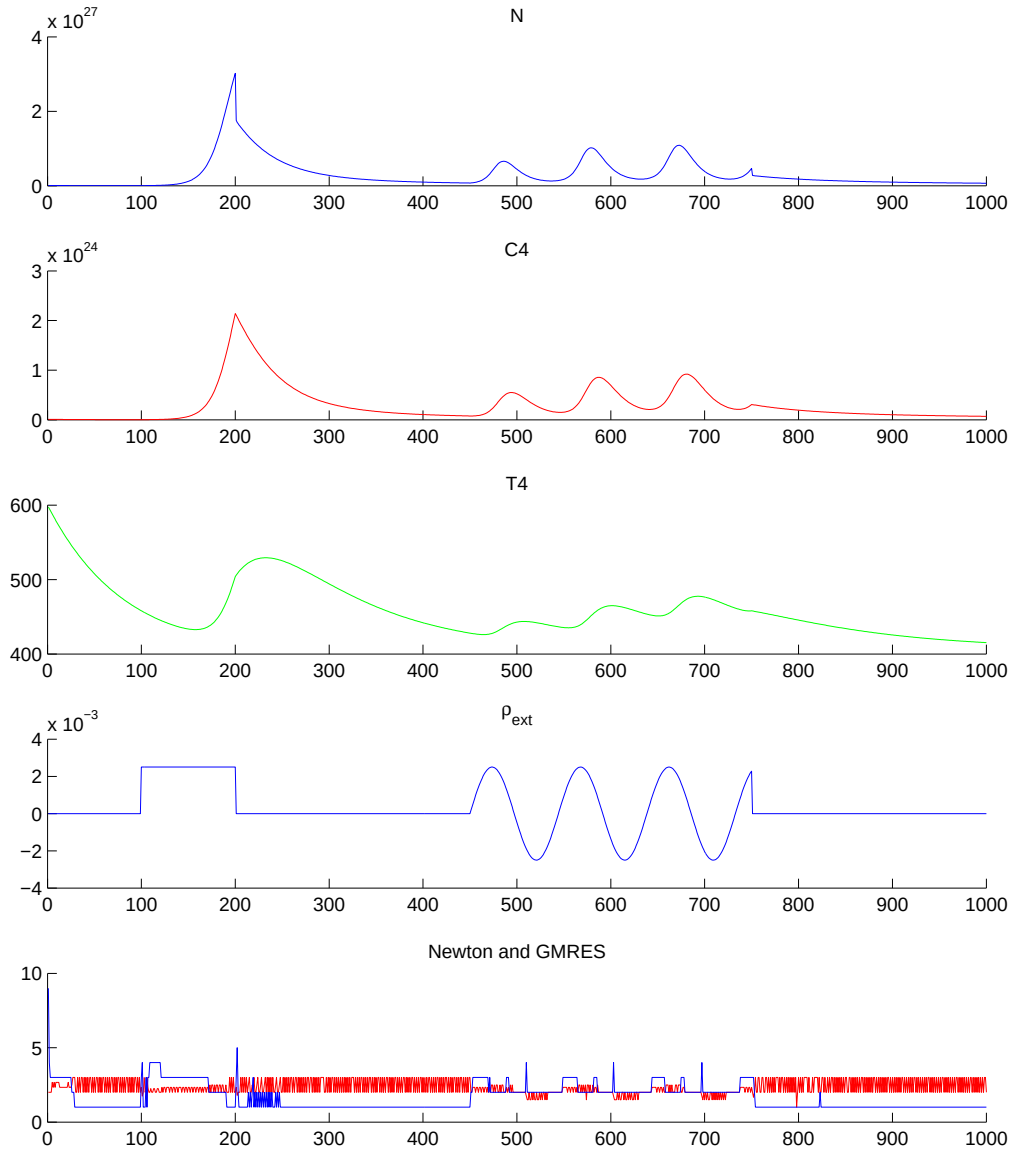


Figure 23: *As function for ρ_{ext} some jumps and a sine function are used to test the model. This function is handled with succes.*

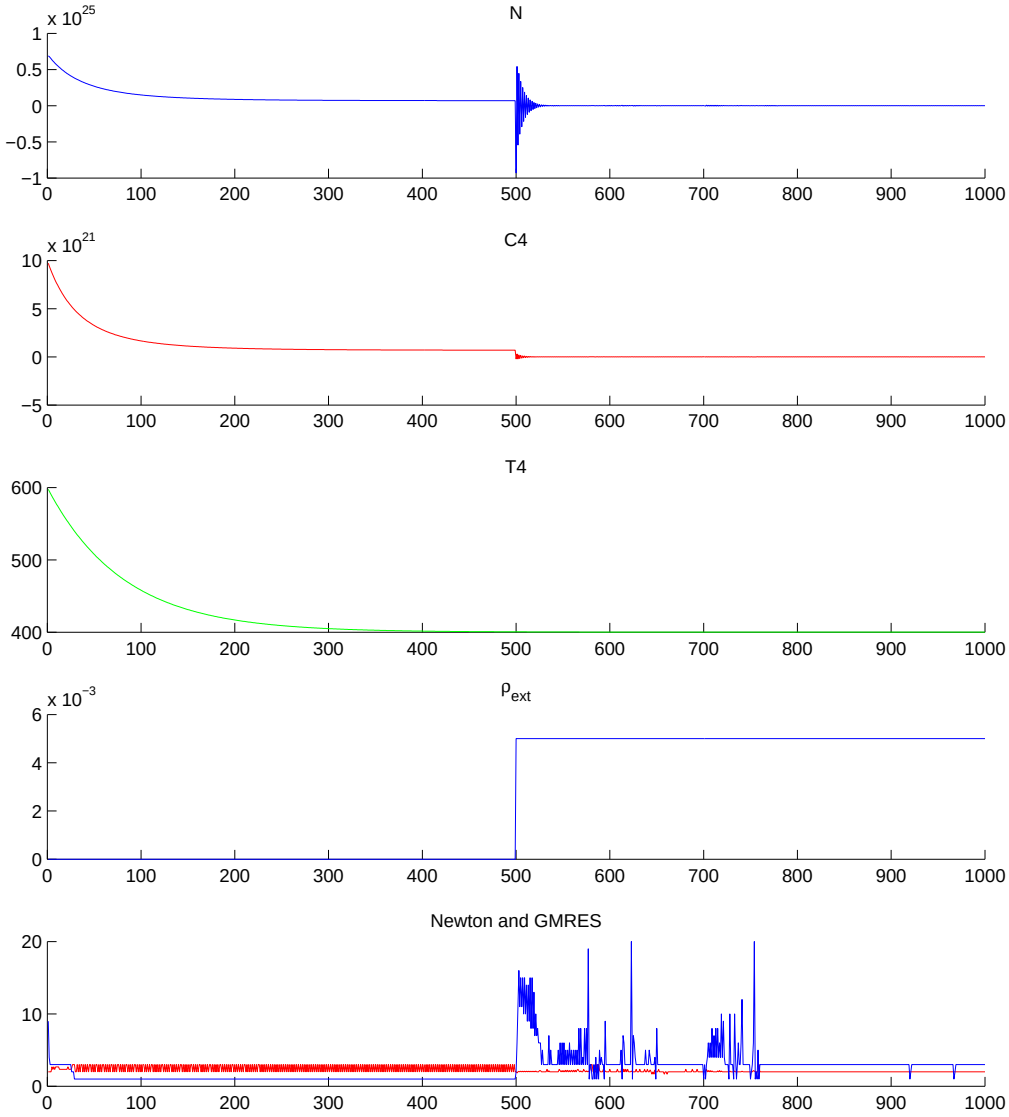


Figure 24: *Jump in the external reactivity of $+\beta$, this causes the method to find a non-relevant solution of the equations (a solution that is not realizable). The number of neutrons in the core becomes negative in this solution. The maximum number of Newton iterations was set to 100, since the algorithm never used more that 20, the Newton loop did converge.*

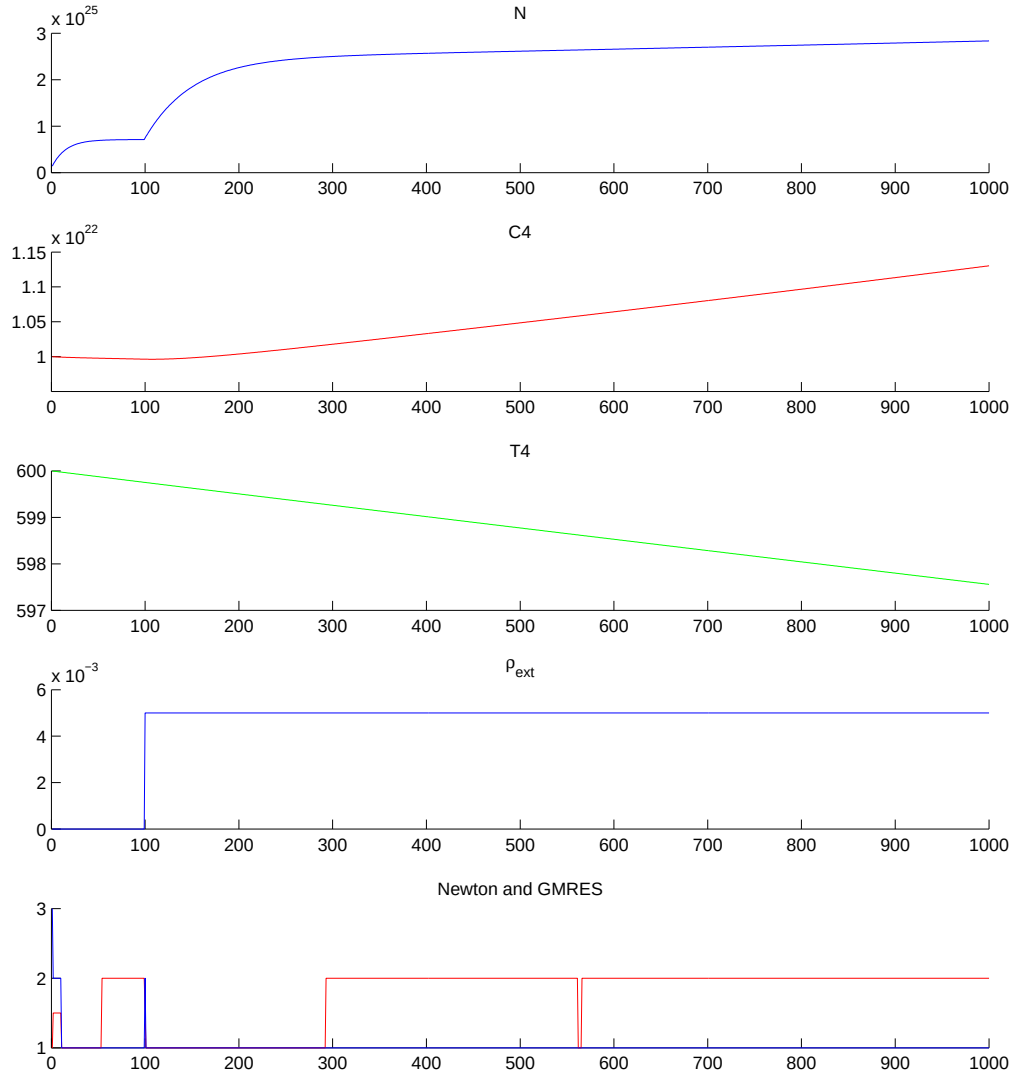


Figure 25: *Solution of the JFNK method of a jump of $+\beta$ in the external reactivity, with a small time step ($\Delta t = 10^{-3}$s). This produces the right solution, as opposed to a solution with a larger time step.*

variables can be too far apart. Relative variables were also discussed in the mathematical theory of JFNK section.

4.2.3 Version 3: Complete model

Version three is the same as version except for the pump, in this version the pump is ”switched on”. This means the governing equations were adapted for use with the pump, all terms with the flow-rate in it

($g(z,t)$) were previously neglected. The test of this version consists of pumping around precursors. An additional test is performed by measuring the total number of precursors when they are being pumped around, when the decay is set to zero ($\lambda = 0$), this will be a constant. The model is also tested by letting the reactor reach an equilibrium state starting with a non-equilibrium state.

Figure 26 shows a spatial and temporal plot of the test where precursors are being pumped around. The reactor starts with a high concentration of precursors in cells 75 through 100. While the fuel-salt mixture is being pumped around the precursors move through the system and dissipate throughout the reactor. Figure 27 shows the same solution, with a plot of the total amount of precursors, which should be a constant. This plot also shows the precursor density and temperature of the fourth cell. In this test not many Newton iterations are needed to solve the problem, only one or two. Apparently this is easier to solve than a problem with a non-zero external reactivity. There were also few GMRES iterations needed to solve the problem.

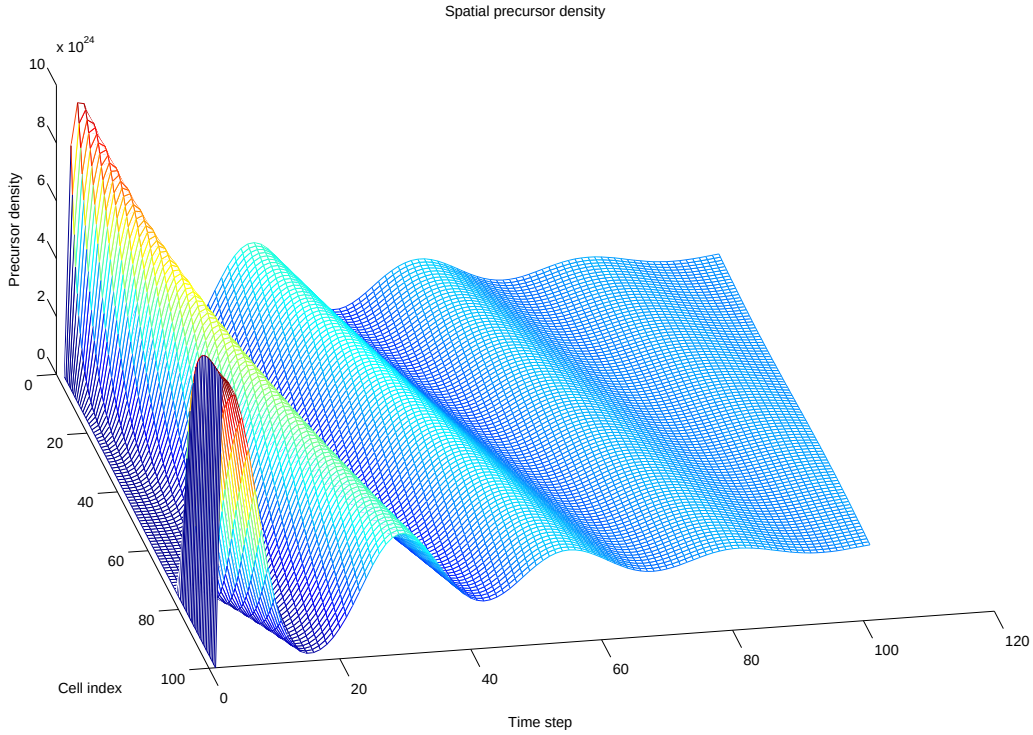


Figure 26: *Spatial plot of the test where precursors are pumped around the reactor, decay of the precursors is not accounted for in this test ($\lambda = 0$), as to test whether the model takes into account the diffusion of the precursors. The fuel-salt mixture flows from cells with lower index to cells with higher index.*

The final test consists of letting the MSR reach an equilibrium state starting with a non-equilibrium state. This shows the model works when using a non-zero flow rate. The plot of this simulation can be found in Figure 28. Notice that very few Newton iterations are needed. Mostly just one Newton iteration, that is much less than the problems where the external reactivity is non-zero. Also the test where precursors are just being pumped around is easy, it also uses mostly one Newton iteration. As to put JFNK to the test, one can best use a problem where the external reactivity is non-zero.

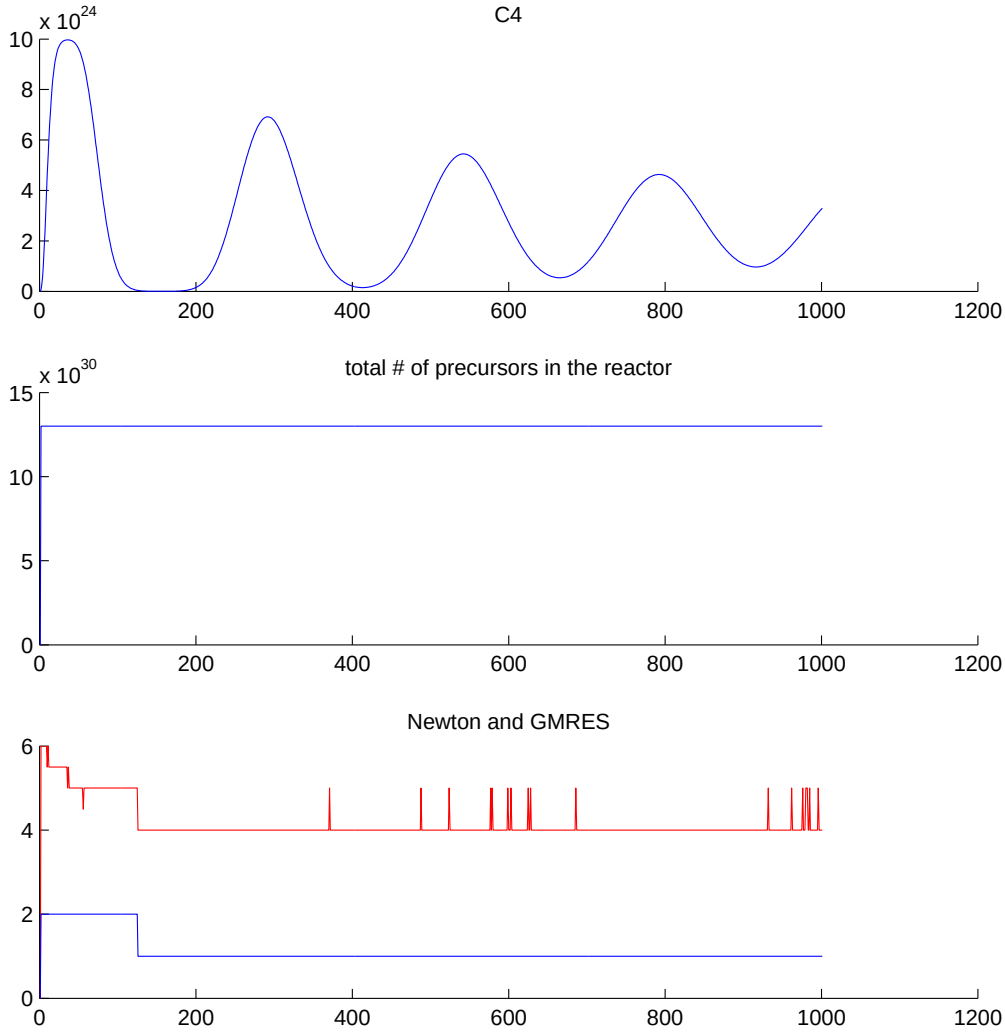


Figure 27: *Same solution as in Figure 26, with a plot of the total number of precursors in the reactor. This should be a constant, as precursor decay is "switched off" ($\lambda = 0$). In this test hot fuel-salt mixture was pumped around as well, as can be deduced from the plot of the temperature of cell 4. There is an initial jump in the plot because Matlab cannot plot a straight horizontal line from data points, and has nothing to do with the solution.*

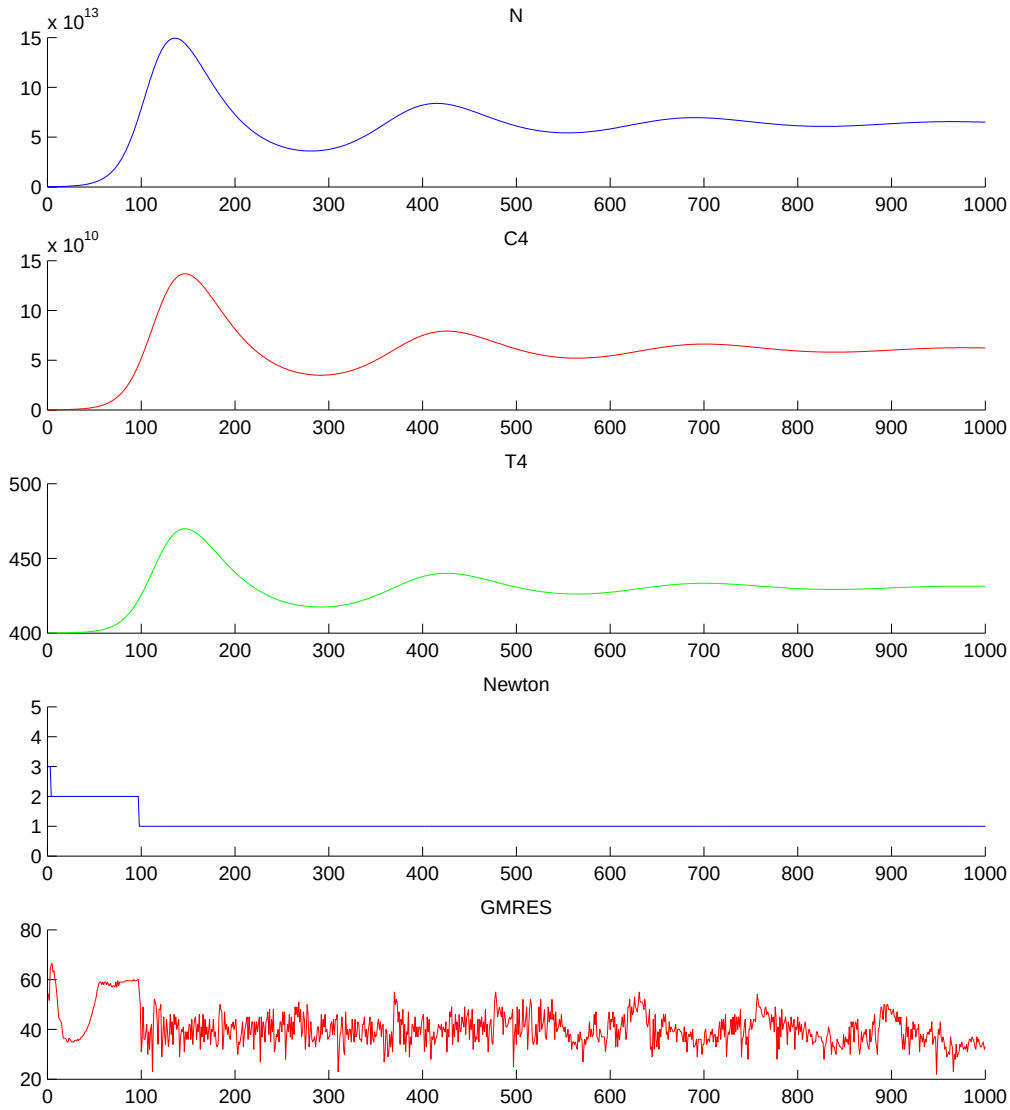


Figure 28: *Test of the final model of the MSR, the reactor reaches an equilibrium state, starting from a non-equilibrium state. This solution is with no external reactivity. The GMRES plot is average number of GMRES iterations per Newton iteration.*

5 Conclusions and discussion

We have studied JFNK methods, or more precise the application of JFNK methods in nuclear reactor physics. In this project is may be concluded JFNK has many applications in nuclear reactor physics. Because many reactors have governing equations of different areas of physics. This is exactly the strength of JFNK, it can solve coupled problems quickly. The full potential of JFNK has not been discovered in this project, since not everything about JFNK has been investigated. Also the test problems were not too difficult for the JFNK method, the harder problems where JFNK could break down have not been found.

In this research two test problems were used, the first was a 1-D rod which was heated on both ends and radiated heat away in the radial direction. In this test problem the differences between "ordinary" methods and the JFNK method were investigated. The second test problem is a simulation of a Molten Salt Reactor (MSR), this problem is harder to solve than the 1-D rod, since different "kinds" of physics are involved in the same problem. This is an aspect the JFNK method is supposed to handle well, this test problem is used to find that out.

Three different methods of the same model of the 1-D rod were used. Just a Newton iteration with a solver for the matrix equation (method I), a Newton iteration with GMRES to solve the matrix equation without the JFNK approximation of the Jacobian (method II), and last a Newton iteration with GMRES and with the JFNK approximation (method III). The time it took to run the simulation of methods II and III was compared. In general method III was faster than method II, except for some cases in which the tolerance of the error was extremely small.

Also the coupling of two systems was tested using the first test problem. In this test the rod is divided into two parts, which can only "see" each other through the neighbouring value of the temperature. In effect this means solving both halves as if it is one rod. The difficulty is that by changing the solution of one part, affects the temperature at the divide, and therefore the solution of the other part. This coupling was done in three ways, two where values on the divide are exchanged (after each Newton loop or after each GMRES loop) and the JFNK method.

All three method produce the same solution when the tolerances are tight enough. The JFNK method finished the calculation more than twice as fast as the other methods, because in total less iterations are used. Also there is not a measure of the error in the solution in the ordinary coupling methods. A situation where this causes problems is found in the ordinary coupling methods. The JFNK method does have a measure for the error in the solution, and therefore has a better stop criterion. The JFNK method does calculate the right solution in the case where the ordinary coupling methods fail.

The second test problem, the MSR, is constructed step by step, meaning there are three different versions. The first is made using just point kinetics equations. The second has the spatial configuration of an MSR but the fuel-salt mixture is not being pumped around. In the third method the pump is "switched on", this is the complete model. All three methods work well, when used with scaled variables. The method produces wrong solutions when the numbers it uses to calculate are "too far apart". All numbers are stored as a floating point number, therefore if the exponent of a number is much smaller than that of the other, addition of the two numbers might result in neglecting the smaller of the numbers. This will in turn give rise to errors like a negative number of neutrons or a negative temperature.

The point kinetics model is tested by comparing the solution to a model which does not make use of JFNK. The second model, with temperature feedback and has the spatial configuration, is tested in different situations. This model is tested qualitatively, like a test where the heat produced in the reactor core cannot flow out of the core. In this case the reactor should shut down, which it does in this model. The third model uses a pump to pump around the fuel-salt mixture. This model is tested by pumping around precursors, without letting them decay. This should result in a high concentration of precursors flowing around the system, while dissipation spreads them through the reactor. The total number of

precursors is also tracked, since the total number should not change when decay is not taken account of in the model. The full model is also tested with a situation in which the pump is started, which would be the start up of the reactor.

In conclusion, Jacobian-Free Newton Krylov methods are good methods to find a solution of systems with different parts of physics involved. It solves the problem quicker than the "standard" methods. It solves the problem without a concentration of the error, avoiding incorrect solutions. And it is applicable to all problems of which the governing equations are known, not necessarily the Jacobian matrix of the system. Which allows solving many more problems.

5.1 Future work

The most promising aspect that is left out of this research due to a lack of time is preconditioning. With using preconditioning without JFNK one simplifies the matrix equation (the set of governing equations) by multiplying both sides of the equation by the same matrix **P**. This matrix **P** is constructed in such a way that the equation will be easier to solve, in practice this means transforming the equation $\mathbf{Ax} = \mathbf{b}$ into $\mathbf{PAx} = \mathbf{Pb}$. In this equation the matrix **PA** looks more like the identity matrix than the matrix **A** and therefore the equation is easier to solve.

However, when using JFNK, there is no matrix **A**, since the matrix vector product is approximated in JFNK. Therefore the exact workings of preconditioning will be different when used with JFNK. There are many different ways of using preconditioning. One of the most popular methods is "physics based" preconditioning, where in essence each part of physics in the problem is solved independent of the others. Before solving the whole coupled set of equations.

The exact workings of preconditioning are not part of this report.

References

[1] C.G. Broyden and M.T. Vespucci. *Krylov Solvers for Linear Algebraic Systems*. Elsevier, 2004.

[2] P. Deuflhard. *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms*. Springer, 2004.

[3] J.J. Duderstadt and L.J. Hamilton. *Nuclear Reactor Analysis*. John Wiley and Sons, 1976.

[4] T.H.J.J. van der Hagen H. van Dam and J.E. Hoogenboom. Nuclear reactor physics, lecture notes, April 2005.

[5] C.T. Kelley. *Solving Nonlinear Equations with Newton's Method*. SIAM, 2003.

[6] D.A. Knoll and D.E. Keyes. Jacobian-free newton-krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, August 2008.

[7] T.M. Merlis and S. Khatiwala. Fast dynamical spin-up of ocean general circulation models using newton-krylov methods. *Ocean Modelling*, 2007.

[8] J.C. Ragusa and V.S. Mahadevan. Consistent and accurate schemes for coupled neutronics thermal-hydraulics reactor analysis. *Nuclear Engineering and Design*, 2008.

[9] Tony F. Chan James Demmel June M. Donato Jack Dongarra Victor Eijkhout Roldan Pozo Charles Romine Richard Barrett, Michael Berry and Henk Van der Vorst. Templates for the solution of linear systems: Building blocks for iterative methods.

[10] D.A. Knoll S.Y. Kadioglu and C. de Oliveira. Analysis and numerical solution for multi-physics coupling of neutron diffusion and thermomechanics in spherical fast burst reactors. *American Nuclear Society*, 2009.

A Nomenclature

$\mathbf{u}$	State vector of model	
$\delta\mathbf{u}$	Step of Newton iteration	
$\mathbf{F}$	Vector function, JFNK finds its roots	
ϵ	Parameter for Jacobian approximation	
k	Index for Newton loop	
n	Index for time	
i	Index for position	
$\mathbf{J}$	Jacobian matrix	
τ	Tolerance for stop criteria	
$\mathbf{v}$	Vector in GMRES, in matrix vector product with $\mathbf{J}$	
T_{left} and T_{right}	Boundary conditions in first test problem	K
Δx	Spatial step	cm
Δt	Temporal step	s
L	Length of rod in first test problem	cm
k	Thermal conductivity in first test problem	W/mK
σ	Stefan-Boltzmann constant	$5.67 \cdot 10^{-8}$ W/cm^{2}K^4
N	Number of neutrons in the reactor	#
C	Precursor density in the reactor	# /cm^{-3}
ρ	Reactivity of the reactor core	
ρ_{ext}	External reactivity	
β	Delayed neutron fraction	
Λ	Neutron generation time	s
λ	precursor decay constant	1/s
A	Cross sectional area	cm^2
ϕ	Adjoint flux	
f	Fission shape	1/cm
g	Flow rate of fuel-salt mixture	cm^3/s
p_{fiss}	Average energy per fission	$3 \cdot 10^{-11}$ J/fission
ν	Average generation of neutrons per fission	2.5 # /fission
(ρc_p)	Heat per volume of fuel or fuel-graphite	J/cm^3
T	Temperature of fuel-salt mixture in test problem 2	K
h	Heat exchange coefficient	W/cm^{2}K
O	Circumference of heat exchanger	cm
T_{he}	Temperature of heat exchanger coolant	K
α	Temperature feedback coefficient	1/K
T_0	Equilibrium temperature of reactor	K